L. M. DEL PINO GONZÁLEZ

REALIDAD VIRTUAL

editorial Paraninfo sa

1995

© LUIS MANUEL DEL PINO GONZÁLEZ
© EDITORIAL PARANINFO, S.A.
Magallanes, 25 - 28015 Madrid
Teléfono: 4463350 - Fax: 4456218
Diseño de cubierta:
© Montytexto, S.L.

Reservados los derechos para todos los países. Ninguna parte de esta publicación, incluido el diseño de la cubierta, puede ser reproducida, almacenada o transmitida de ninguna forma, ni por ningún medio, sea éste electrónico, químico, mecánico, electro-óptico, grabación, fotocopia o cualquier otro, sin la previa autorización escrita por parte de la Editorial.

Impreso en España
Printed in Spain

ISBN: 84-283-2152-3

Depósito Legal: M. 2.805-1995

editorial Paraninfo sa Magallanes, 25 - 28015 MADRID (012/51/09)

Gráficas ROGAR, Polígono Industrial Cobo Calleja - Fuenlabrada (Madrid)

A Mª José, Álvaro y Natalia, que me han dejado ser una realidad casi virtual durante el tiempo que he tardado en escribir este libro.

A mi madre, a quien también he robado más tiempo de lo que me hubiera gustado.

A Mercedes y a Pepe, que siempre están ahí para hacer más sencillas las cosas.

Índice

PRÓLOGO ..... **XV**

AGRADECIMIENTOS ..... **XVII**

1. SISTEMAS DE REALIDAD VIRTUAL **1**

1.1. UN EJEMPLO PARA EMPEZAR 1

1.1.1. Estructura del sistema 1

1.1.2. Funcionamiento de la aplicación 2

1.1.3. El sistema de realidad virtual por dentro 5

1.2. TIPOS DE SISTEMAS DE REALIDAD VIRTUAL 6

1.2.1. Sistemas inmersivos 6

1.2.2. Sistemas proyectivos 6

1.2.3. Sistemas de sobremesa 7

1.3. APLICACIONES 7

1.3.1. Prototipado 7

1.3.1.1. Diseño industrial 8

1.3.1.2. Diseño arquitectónico 9

1.3.1.3. Planificación de operaciones 9

1.3.1.4. Paseos virtuales 9

1.3.2. Simulación 10

1.3.3. Aplicaciones metafóricas 11

1.3.4. Tele-robótica 12

1.3.5. Aplicaciones de ocio 13

1.3.5.1. Juegos de galería 13

1.3.5.2. Juegos inmersivos 15

1.3.5.3. Juegos de plataforma 16

2. PANORÁMICA GENERAL **19**

2.1. SISTEMAS DE RV: QUÉ SON 19

2.1.1. Definición 19

2.2. CARACTERÍSTICAS DE LOS SISTEMAS DE RV 20

2.2.1. Capacidad sintética 20

2.2.2. Interactividad 21
2.2.3. Tridimensionalidad 24
2.2.4. Ilusión de realidad 25
2.2.4.1. Factores físicos 26
2.2.4.2. Factores psicológicos 26
2.3. MECANISMOS BÁSICOS 27
2.3.1. Gráficos tridimensionales 27
2.3.2. Técnicas de estereoscopía 28
2.3.3. Simulación de comportamiento 28
2.3.4. Facilidades de navegación 30
2.3.5. Técnicas de inmersión 32
2.4. CARACTERIZACIÓN DE LOS SISTEMAS DE RV 33
2.5. ARQUITECTURA HARDWARE 34
2.5.1. Diagrama general 34
2.5.2. Dispositivos de entrada 34
2.5.3. Dispositivos de salida 35
2.5.4. Estación de proceso 36
2.6. ALGUNOS PARÁMETROS DE IMPORTANCIA 37
2.6.1. Velocidad de generación de las imágenes 37
2.6.2. Latencia 38
2.6.3. Compromisos 40
2.7. CLAVES DE PROFUNDIDAD 41
2.7.1. Claves físicas 41
2.7.1.1. Disparidad binocular o estereoscopía 41
2.7.1.2. Paralaje 41
2.7.1.3. Convergencia 42
2.7.1.4. Acomodación 43
2.7.1.5. Limitaciones de los sistemas de RV 43
2.7.2. Claves psicológicas 43
2.7.2.1. Perspectiva lineal 44
2.7.2.2. Interposición 44
2.7.2.3. Sombreados y sombras 44
2.7.2.4. Gradiente de textura 45
2.7.2.5. Difuminación 45
3. DISPOSITIVOS DE SALIDA 47
3.1. DISPOSITIVOS DE PRESENTACIÓN 48

3.1.1. Tipos de dispositivos 48
3.1.1.1. Sistemas inmersivos 49
3.1.1.2. Sistemas proyectivos y de sobremesa 50
3.1.1.3. Técnicas de estereoscopía 51
3.1.1.4. Problemas de los sistemas estereoscópicos 51
3.1.2. Visiocascos 52
3.1.2.1. Características técnicas 52
3.1.2.2. Problemas de los visiocascos 54
3.1.2.3. Sistemas comerciales existentes 55
3.1.3. Sistemas binoculares 56
3.1.3.1. Ventajas y desventajas de los sistemas binoculares 57
3.1.3.2. Sistemas comerciales existentes 57
3.1.4. Gafas estereoscópicas 57
3.1.4.1. Aliasing temporal 58
3.1.4.2. Sistemas comerciales 59
3.1.5. Sistemas autoestereoscópicos 59
3.2. DISPOSITIVOS DE AUDIO 59
3.2.1. El sonido en las aplicaciones de RV 59
3.2.2. Creación de las secuencias sonoras 60
3.2.2.1. Reproducción de secuencias pregrabadas 61
3.2.2.2. Síntesis de sonido 61
3.2.3. Sonido 3-D 62
3.2.4. Reproducción del sonido 64
3.2.5. Sintetizadores de voz 65
3.3. OTROS DISPOSITIVOS DE SALIDA 66
3.3.1. Plataformas móviles 66
3.3.2. Dispositivos táctiles 67
3.3.2.1. Dispositivos de realimentación táctil 68
3.3.2.2. Dispositivos de realimentación cinestésica 69
4. DISPOSITIVOS DE ENTRADA 71
4.1. DISPOSITIVOS DE LOCALIZACIÓN 72
4.1.1. Funciones 72
4.1.1.1. Detección de la orientación del usuario 72
4.1.1.2. Detección de la posición del usuario 73
4.1.1.3. Control de la aplicación 73
4.1.1.4. Digitalización de objetos 74
4.1.2. Dispositivos electromagnéticos 74

4.1.2.1. Sistemas comerciales 76
4.1.3. Dispositivos mecánicos 77
4.1.4. Dispositivos ultrasónicos 78
4.1.5. Otros sistemas de localización 78
4.2. DISPOSITIVOS DE CONTROL 79
4.2.1. Electroguantes 80
4.2.1.1. Aplicaciones 80
4.2.1.2. Tecnologías existentes 82
4.2.1.3. Sistemas comerciales 83
4.2.2. Ratones y joysticks 3-D 83
4.2.3. Dispositivos para simuladores 84
4.2.4. Sistemas bioeléctricos 84
4.3. OTROS DISPOSITIVOS DE ENTRADA 86
5. SÍNTESIS DE IMÁGENES 3-D 87
5.1. EL PROCESO DE SÍNTESIS (I) 88
5.2. GEOMETRÍA DE LAS ESCENAS 89
5.2.1. Definición geométrica de los objetos 89
5.2.2. Definición geométrica de las escenas 90
5.3. MODELOS DE ILUMINACIÓN 92
5.3.1. Sombreado aleatorio 93
5.3.2. Sombreado plano 94
5.3.3. Sombreado de Gouraud 97
5.3.4. Sombreado de Phong 99
5.3.5. Un modelo real 99
5.4. EL PROCESO DE LA SÍNTESIS 100
5.4.1. Transformación de coordenadas 101
5.4.2. Clipping 102
5.4.3. Cálculo de la iluminación 103
5.4.4. Proyección 103
5.4.5. Z-buffering 104
5.4.6. Sombreado 108
5.4.7. Presentación de la imagen 108
5.4.8. El algoritmo de síntesis al completo (versión simple) 110
5.5. OPTIMIZACIÓN DEL PROCESO DE SÍNTESIS 112
5.5.1. Selección (culling) 113
5.5.2. Eliminación de caras ocultas 116

5.5.3. Texturado 117
5.5.4. Bitmaps auto-orientables 122
5.5.5. Otras técnicas 123
5.6. VARIACIÓN DINÁMICA DEL NIVEL DE DETALLE 124
5.6.1. Variación del nivel de detalle con la distancia 124
5.6.2. Variación del nivel de detalle con la velocidad 125
5.6.3. Variación del nivel de detalle con el foco de atención 125
5.6.4. Variación del nivel de detalle por selección del usuario 126
5.6.5. Variación del nivel de detalle con la carga de trabajo 126
5.7. EL PROCESO DE SÍNTESIS (II) 126
5.7.1. El proceso de síntesis revisitado 126
5.7.2. Problemas del proceso de síntesis 127
5.7.3. Anti-aliasing 128
5.7.3.1. Sobremuestreo 129
5.7.3.2. Filtrado 129
5.7.3.3. El problema del aliasing en las texturas 131
5.7.4. ¿Cómo definir un objeto de manera perfecta? 132
6. ESTACIÓN DE PROCESO 135
6.1. FUNCIONES 135
6.1.1. La tarea de simulación 136
6.1.2. Animación 139
6.1.2.1. Objetos animados e inanimados 139
6.1.3. Objetos simples y compuestos 141
6.2. ARQUITECTURA HARDWARE 142
6.2.1. Arquitectura básica 142
6.2.2. Arquitecturas gráficas avanzadas 143
6.2.2.1. Arquitecturas pipeline 143
6.2.2.2. Arquitecturas paralelas 144
6.2.2.3. Un ejemplo de arquitectura combinada 145
6.2.3. Arquitecturas distribuidas 146
6.3. TIPOS DE ESTACIONES 148
6.3.1. Estaciones especializadas 149
6.3.2. Estaciones gráficas 149
6.3.3. Estaciones de propósito general 150
6.4. ARQUITECTURA SOFTWARE 152
6.4.1. Diagrama general 152

6.4.2. Librerías gráficas 153
6.4.3. Entorno de ejecución 154

7. DESARROLLO DE APLICACIONES DE RV 157

7.1. CÓMO SE CREA UN PROGRAMA DE RV 157
7.2. ESPECIFICACIÓN DE REQUERIMIENTOS 158
7.3. DISEÑO E INTEGRACIÓN DE LA PLATAFORMA HW/SW 162
7.4. DISEÑO GRÁFICO DE LOS OBJETOS 164
7.4.1. Pasos del proceso 164
7.4.2. Modelado de objetos 3-D 166
7.4.2.1. Métodos constructivos 166
7.4.2.2. Métodos algorítmicos 167
7.4.3. Animación en 3-D 167
7.4.4. Generación de la base de datos gráfica 168
7.4.5. Programas de modelado y animación comerciales 169
7.4.6. Técnicas y programas de digitalización 171
7.4.7. Formatos gráficos 171
7.5. ENTORNOS DE DESARROLLO 172
7.5.1. Funciones 172
7.5.2. Worldtoolkit 173
7.5.2.1. Descripción del producto 173
7.5.2.2. Versiones 174
7.5.2.3. Periféricos soportados 175
7.5.3. Superscape 175
7.5.3.1. Descripción del producto 175
7.5.3.2. Opciones 176
7.5.3.3. Periféricos soportados 177
7.5.4. dVISE 177
7.5.4.1. Descripción del producto 177
7.5.4.2. Versiones existentes 177
7.5.4.3. Periféricos soportados 178
7.5.5. Otros sistemas de desarrollo 178
7.5.5.1. VREAM 178
7.5.5.2. Cyberspace Developer Kit 179
7.5.6. El problema del comportamiento 179

8. EL FUTURO DE LA REALIDAD VIRTUAL 181

8.1. UN POCO DE HISTORIA 182

8.2. LA REALIDAD VIRTUAL HOY 187
8.2.1. Panorama del sector 187
8.2.2. La RV en España y en Europa 188
8.3. LÍNEAS PRINCIPALES DE INVESTIGACIÓN 190
8.3.1. Periféricos 190
8.3.1.1. Visiocascos 190
8.3.1.2. Dispositivos de localización 191
8.3.1.3. Electroguantes 191
8.3.1.4. Otros dispositivos 192
8.3.2. Dispositivos gráficos 192
8.3.3. Paradigmas de control 193
8.3.4. Comportamiento e interacción 193
8.3.5. Sistemas de RV en red 195
8.4. APLICACIONES DE MAÑANA 195
8.4.1. Aplicaciones médicas 195
8.4.2. Sistemas de ocio 196
8.4.3. Sistemas de entrenamiento y simulación 197
8.5. RV EN RED Y CIBERESPACIOS 197
8.5.1. Sistemas de RV en red 197
8.5.1.1. Ciberespacios 197
8.5.1.2. Sistemas multiparticipante 198
8.5.1.3. Sistemas distribuidos 198
8.5.1.4. Sistemas en red 198
8.5.2. Implementación de los sistemas de RV en red 199
8.5.2.1. Arquitecturas de baja latencia 199
8.5.2.2. Arquitecturas de alta latencia 202
8.5.2.3. Interacción próxima y lejana 203
8.5.3. Algunos ejemplos de ciberespacios actuales 204
8.5.3.1. SIMNET Y NPSNET 204
8.5.3.2. Habitat 205

Prólogo

Cuando comencé a escribir este libro, me propuse contar en él básicamente una cosa: lo que me hubiera gustado que alguien me contara a mí cuando empecé a tomar contacto con el mundo de la realidad virtual.

En consecuencia, he procurado explicar en detalle, pero con términos claros y sencillos, qué es la realidad virtual, cómo funcionan los sistemas de RV, cuáles son las tecnologías que hoy en día se utilizan, cómo se desarrolla una aplicación de RV, qué productos comerciales existen, cuáles son las principales áreas de aplicación, qué relación guarda la RV con otros campos y cuáles son las tendencias futuras del mercado.

El libro es el resultado de más de un año de investigaciones y de recopilación de información; durante este tiempo, algunas secciones han tenido que ser reescritas multitud de veces, ante la avalancha de cambios que día a día se producen. Pero la mayor parte del tiempo ha estado dedicada no a escribir, sino a cribar y analizar la maraña de informaciones disponibles, tratando de ir más allá de la mera descripción de los datos, hasta encontrar la mejor manera de explicar los cómos y los porqués del funcionamiento de un sistema de RV. Frecuentemente, en un mercado naciente como éste, los datos disponibles tienen más de ciencia ficción y de marketing que de verdadera información, por lo que resulta fácil perderse en lo accesorio. Después de leer este libro, espero que usted comprenda algo mejor la filosofía de los sistemas de realidad virtual, y que le sea fácil distinguir lo esencial de lo que sólo es mera literatura.

El libro se abre con un capítulo introductorio, que no tiene otro fin que situar al lector ante un ejemplo de sistema de RV y exponer los principales campos de aplicación de esta tecnología. En el capítulo 2 se proporciona una visión panorámica de los sistemas de RV: sus características, los principales mecanismos en ellos utilizados, los parámetros más importantes que los caracterizan, y los distintos componentes que los forman. Al análisis de estos componentes están dedicados los cuatro capítulos siguientes.

Los capítulos 3 y 4 tratan de los dispositivos periféricos de salida y de entrada, respectivamente. En ellos hablaremos de los diversos tipos de dispositivo más comúnmente asociados con los sistemas de RV, y que tanto han contribuido a popularizar este tipo de tecnología: visiocascos, dispositivos de localización, electroguantes, ratones 3D, ... Para cada tipo de periférico, se exponen las tecnologías existentes, sus aplicaciones, los principales problemas y los productos comerciales más extendidos.

El tema de las estaciones de proceso utilizadas en los sistemas de realidad virtual se trata en el capítulo 6, donde se exponen los principios de funcionamiento, las arquitecturas más comunes, los distintos tipos de plataformas y las principales soluciones existentes en el mercado. Uno de los aspectos más trascendentales del funcionamiento de una plataforma de

RV es el de la generación de gráficos en tiempo real. Dada la importancia y la amplitud del tema, he preferido dedicarle un capítulo entero, el capítulo 5. En él se analiza en detalle el proceso de síntesis de escenas tridimensionales, revisando los principales algoritmos y técnicas existentes, así como los trucos más comúnmente empleados en aplicaciones de tiempo real.

El proceso de desarrollo se trata en el capítulo 7, donde se pasa revista tanto al diseño de sistemas como al desarrollo de aplicaciones. En él se explican los distintos pasos del proceso, incluyendo la etapa de diseño gráfico, y se proporcionan numerosas indicaciones que espero le sean útiles a la hora de implementar cualquier aplicación. En la última sección del capítulo se realiza un repaso de los principales sistemas comerciales de desarrollo de aplicaciones de RV.

El capítulo que cierra el libro adopta una perspectiva esencialmente temporal. En él se incluye una breve referencia histórica de los sistemas de RV, y se exponen las principales tendencias que se apuntan en el mercado, las cuales marcarán el ritmo de los acontecimientos durante los próximos años. Siempre es arriesgado el oficio de profeta, pero creo que las tendencias que se detectan son bastante claras para cualquiera que siga la evolución de esa amalgama que hoy se conoce con el nombre de mercado de la multimedia.

Si alguien me preguntara qué objetivo me gustaría cubrir con el libro, creo que lo que haría sería llamarle la atención sobre las distintas compañías que a lo largo del libro se mencionan. La práctica totalidad de las compañías que hoy dominan el sector son de creación relativamente reciente (veinticinco años, a lo sumo), pese a lo cual muchas de ellas son hoy poderosas empresas multinacionales. Como cualquier otro mercado naciente, la realidad virtual es un mercado de riesgo, donde es fácil naufragar, pero donde las expectativas de beneficio para el que sobrevive son inmensas. Si este libro contribuye a que uno sólo de sus lectores se decida a investigar, trabajar o invertir en el campo de la realidad virtual, el autor considerará que su objetivo se ha cumplido.

Agradecimientos

Muchas son las personas que me han ayudado durante la preparación de este libro, proporcionándome información acerca de distintos aspectos del mundo fascinante de la realidad virtual. Gracias a Salvador Bayarri, del LISITT; Alberto Arnedillo, de Realidad Virtual S.L.; y Roger Gilabert, de RTZ Technologies, quienes se cuentan entre el pequeño grupo de pioneros de la realidad virtual en España. Gracias también a Christian Dutilleux, de Deltatec, y a Ludwig Möhlmann, de Nova Apparate, cuyo conocimiento del mercado de los juegos sólo es superado por su sentido práctico.

Gracias muy especialmente a Patrick Leclerc y Pascal Cottin, de Caption, y a Pascual Caselles, de APD, quienes me han enseñado casi todo lo que sé sobre generación de gráficos en tiempo real.

Algunos de los aspectos más avanzados de la realidad virtual, como todo lo relativo a las arquitecturas distribuidas, o las tendencias en programación de aplicaciones de simulación, no hubieran podido ser tratados en este libro de no ser por las interminables horas de discusión con Dag Belsnes y Brynjulv Hauksson, del Norsk Regnesentral, y con el equipo del Departamento de Sistemas Distribuidos del Trinity College de Dublin: Neville Harris, Brendan Tangney, Vinny Cahill, Andrew Condon y John McLaughlin. A todos ellos mi gratitud. Gracias asimismo a Richard Watson, también del Trinity College, por la información sobre electroguantes y lenguajes gestuales.

Y finalmente, mi más absoluta gratitud a Mª José, mi esposa, quien tuvo la paciencia de leerse el manuscrito y me proporcionó valiosas sugerencias que han contribuido a hacer este libro mucho más ameno; y a Mª Teresa Gómez-Mascaraque, de la Editorial Paraninfo, quien (una vez más) ha hecho gala de su gran paciencia mientras yo me eternizaba escribiendo las páginas que siguen.

1
Sistemas de realidad virtual

1.1. UN EJEMPLO PARA EMPEZAR

La empresa inglesa Virtuality fue la primera en sacar al mercado un sistema inmersivo de realidad virtual para aplicaciones de ocio, del cual se han instalado en todo el mundo centenares de unidades. Algunos lectores habrán tenido la oportunidad de utilizarlo, dado que existen varias máquinas Virtuality instaladas en nuestro país.

Comenzaremos nuestro viaje por el mundo de la realidad virtual presentando un sistema ficticio similar al Virtuality 1000, con el fin de hacernos una primera idea de cómo funcionan los sistemas de realidad virtual.

1.1.1. Estructura del sistema

En nuestro sistema ficticio, el usuario dispone de dos elementos diferentes, con los que puede interaccionar con el sistema (ver figura 1.2):

- Un visiocasco, que el usuario se coloca en la cabeza. El visiocasco le impide al usuario ver el mundo que le rodea. En lugar de ello, posee una pantalla por cada ojo, donde aparecerán las imágenes del mundo virtual. Las imágenes que aparecen en las dos pantallas son ligeramente diferentes, de forma que el efecto es que el usuario puede ver en relieve.
- Un mando con dos botones. Apretando el primero, el usuario se desplazará en la dirección en la que en ese momento esté mirando. El segundo le va a permitir al usuario disparar contra los objetos que le salgan al paso.

Tanto el visiocasco como el mando de control están conectados a un ordenador, que es el encargado de ejecutar la aplicación de realidad virtual. El visiocasco está conectado a dos salidas de video del ordenador, por las que recibe las imágenes que el ordenador genera. El mando de control está conectado a una tarjeta controladora, mediante la que el ordenador detecta cuándo el usuario ha pulsado un botón.

Existe un tercer dispositivo, del que el usuario no es siquiera consciente, pero que el ordenador utiliza: el visiocasco lleva incorporado un sensor de posición que va a servir al ordenador para detectar en qué dirección está mirando el usuario.

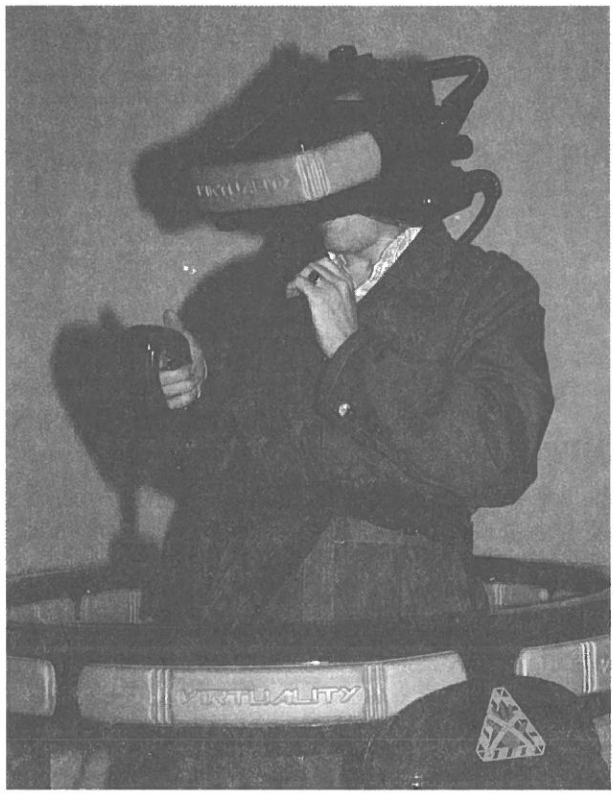

Figura 1.1. Sistema Virtuality. *(Fotografía de Alejandro Sacristán)*

El sensor de posición está conectado a una unidad de control. Dicha unidad calcula la posición del usuario a partir de las medidas que el sensor de posición le envía, y manda esa información al ordenador a través del puerto serie.

1.1.2. Funcionamiento de la aplicación

Cuando el programa se inicia, el usuario se ve inmerso en un mundo de tres dimensiones, en el que hay una serie de objetos que se mueven y contra los cuales tiene que disparar.

El usuario puede ver los objetos moverse porque el ordenador está perpetuamente calculando imágenes y presentándoselas a través del visiocasco. Al calcular las imágenes, el ordenador tiene en cuenta la posición de cada objeto móvil en ese instante, de forma que las imágenes van variando a medida que los objetos se desplazan.

La imagen también varía cuando el usuario mira a un lado y al otro. El sensor de posición adherido al visiocasco permite al ordenador conocer en qué dirección está mirando el usuario en cada momento. Con ese dato, el ordenador calcula la imagen tal como el usuario la vería en caso de encontrarse realmente dentro del mundo virtual. Así, cuando el usuario gira su

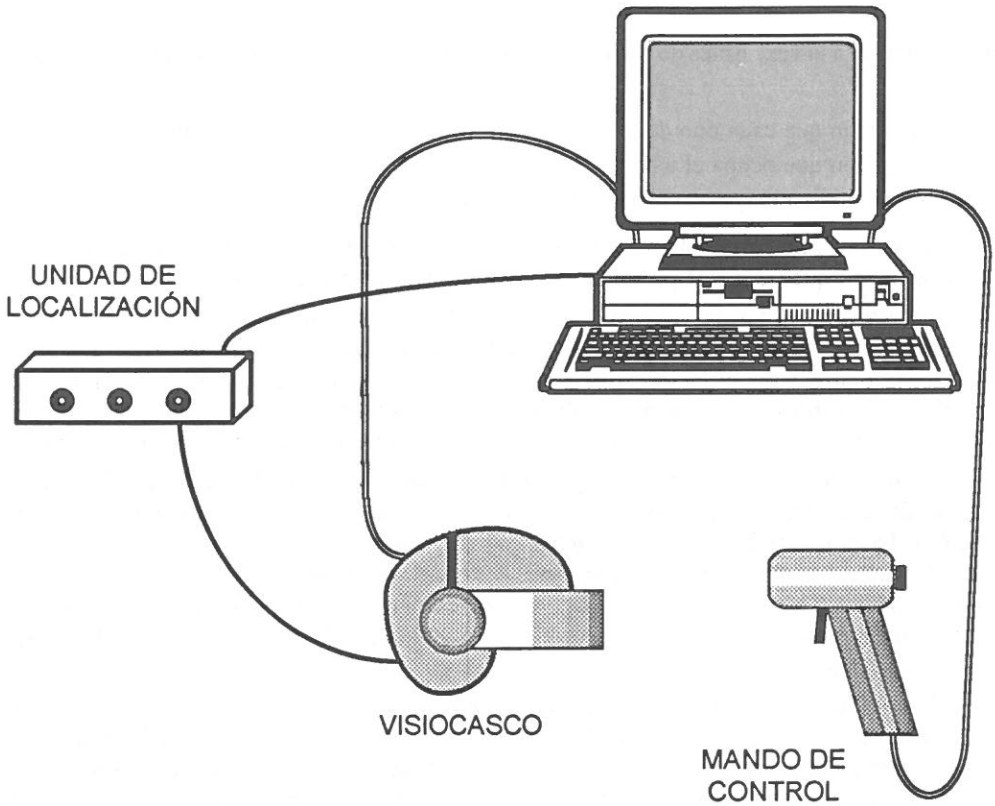

Figura 1.2. Un ejemplo de sistema de RV

cabeza hacia la izquierda, las imágenes en la pantalla se desplazan hacia la derecha, y viceversa.

Para calcular las imágenes, el ordenador no tiene sólo en cuenta la dirección en la que el usuario mira, sino también la posición imaginaria que el usuario ocupa dentro del mundo virtual. De esa manera, el ordenador presenta los objetos próximos al usuario con un mayor tamaño que los objetos más alejados.

Cada vez que el usuario aprieta el botón de desplazamiento, el ordenador modifica la posición imaginaria que ocupa dentro del mundo virtual, desplazándola en la dirección en la que en ese momento esté mirando. Con ello, las imágenes vuelven a variar: los objetos van haciéndose más grandes en pantalla a medida que el usuario se acerca a ellos.

Todas estas variaciones tienen lugar de manera simultánea: los objetos pueden desplazarse en el mundo virtual al mismo tiempo que el usuario se desplaza, y éste puede estar, además, girando la cabeza a un lado y a otro para mirar. El ordenador tiene en cuenta todos

estos factores a la vez. Antes de calcular cada imagen, el ordenador consulta la serie completa de datos:

- posición que cada uno de los objetos ocupa dentro del mundo virtual;
- posición que ocupa el usuario;
- dirección en la que el usuario está mirando;

y, teniendo en cuenta todos ellos, presenta la imagen adecuada. En realidad, las dos imágenes adecuadas, porque tiene que presentar una imagen distinta por cada ojo.

Los objetos del mundo virtual están dotados de un cierto comportamiento. Son capaces de desplazarse cuando tratan de alejarse del usuario que intenta darles caza. En cada momento, la dirección en la que huyen va a venir determinada por la posición del usuario en el mundo virtual. Además de desplazarse, los objetos pueden sufrir los efectos de los disparos que el usuario realiza. Cada vez que un objeto es alcanzado, explota en pedazos.

El ordenador es quien se encarga de simular el comportamiento de los objetos. Para cada objeto de la escena, el ordenador calcula cada vez la acción que va a realizar. Para ello, comprueba en qué posición está el usuario con respecto al objeto, y "toma la decisión" de que el objeto huya en la dirección contraria. A todos los efectos, es como si el propio objeto tomara él mismo la decisión. Esa decisión se traduce en los movimientos que el objeto realiza en la pantalla porque, como resultado de ella, la posición del objeto dentro del mundo virtual va a ir variando.

Pero el ordenador no sólo regula las acciones de los objetos, sino también las interacciones que se pueden producir en el mundo virtual. Cuando el usuario llega a una pared, por ejemplo, el ordenador detiene su marcha, porque hay un obstáculo virtual impidiéndole el paso. Cuando un objeto llega a una pared, el ordenador tiene en cuenta el hecho de que en esa dirección no puede seguir huyendo, y calcula una nueva dirección en la que moverse. De la misma forma, cuando una bala es disparada por el usuario el ordenador traza su recorrido, comprobando si alcanza a alguno de los objetos. Si es así, el objeto explota.

La explosión de los objetos forma parte del comportamiento de éstos. Es su manera de morir. El programador de la aplicación podía haber elegido otro comportamiento menos llamativo, una simple desaparición tal vez, pero prefirió establecer que, al ser alcanzados, explotaran. Al explotar, la forma original del objeto desaparece de la pantalla, y en su lugar aparece una secuencia de imágenes que muestra la explosión del objeto.

Es el ordenador el que genera las imágenes y, por tanto, el que arranca la secuencia de la explosión al detectar que un objeto ha sido alcanzado.

En consecuencia, hay un factor más que el ordenador tiene en consideración a la hora de generar las imágenes del mundo virtual. Dichas imágenes no dependen sólo de la posición de los objetos y del usuario, y de la dirección en que el usuario mira, sino que también pueden variar dependiendo del comportamiento de los objetos y del estado en que se encuentren. A medida que el objeto ejecuta determinadas acciones, el ordenador puede arrancar distintas secuencias de animación asociadas a las mismas. Dichas secuencias permiten al usuario conocer lo que un objeto hace en cada instante.

1.1.3 El sistema de realidad virtual por dentro

Básicamente, el hardware necesario para ejecutar este tipo de aplicación se reduce a una plataforma convencional (un 486 a 33 MHz, por ejemplo), con un par de tarjetas gráficas para la generación de las señales de video que alimentarán al visiocasco, una tarjeta controladora para el mando y un puerto serie para la comunicación con el controlador del sensor de posición que el visiocasco lleva adherido.

En cuanto al software, la estructura de nuestra aplicación es también bastante sencilla. El ordenador está continuamente ejecutando un bucle infinito, hasta que la partida termina:

PASO 1 Leer los datos de los dispositivos de entrada

- Leer el estado del mando de control, para ver si el usuario quiere desplazarse o ha disparado.
- Leer la posición calculada por el dispositivo de localización, para averiguar hacia dónde está mirando el usuario.

PASO 2 Realizar la simulación del mundo virtual

- actualizar la posición del usuario
- actualizar la posición de los otros objetos
- si algún objeto ha sido alcanzado por los disparos, iniciar la secuencia de explosión correspondiente

PASO 3 Pintar la imagen del mundo virtual correspondiente a la posición y orientación que en ese momento tiene el usuario

Como puede verse, la estructura no puede ser más simple (figura 1.3):

Esta simplicidad no se debe sólo al hecho de que nuestro ejemplo sea sencillo. A lo largo de los próximos capítulos veremos que esta estructura básica (con una u otra variante)

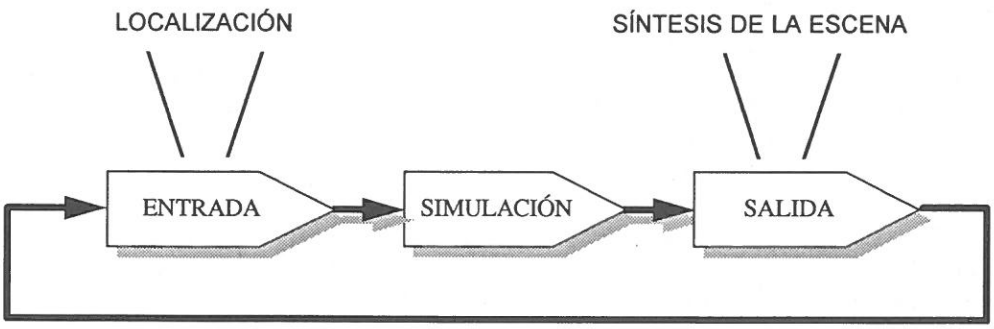

Figura 1.3. Estructura de una aplicación de RV

describe casi perfectamente el funcionamiento de la mayoría de las aplicaciones de RV comerciales.

Las aplicaciones de RV son complejas, pero esta complejidad no radica en su estructura, sino en la propia dificultad que cada uno de esos tres pasos básicos conlleva. Porque las aplicaciones de RV utilizan la más avanzada tecnología de dispositivos de entrada, las más desarrolladas técnicas de simulación y los más complejos algoritmos y equipos para presentar al usuario la información sobre el mundo virtual en que se encuentra situado.

1.2. TIPOS DE SISTEMAS DE REALIDAD VIRTUAL

Existen tres tipos básicos de sistemas de realidad virtual, dependiendo de la forma en que se presente la información visual: inmersivos, proyectivos y de sobremesa.

1.2.1. Sistemas inmersivos

Los sistemas inmersivos tienen por objeto conseguir que el usuario tenga la sensación de estar realmente "dentro" del mundo virtual representado. Para ello, se utilizan dispositivos que impiden la visión del mundo circundante, al mismo tiempo que presentan las imágenes correspondientes al mundo virtual. El ejemplo más conocido, aunque no el único, es el de los visiocascos.

Por sus propias características, los sistemas inmersivos requieren de algún dispositivo que permita detectar los movimientos de la cabeza del usuario, con el fin de que la imagen corresponda siempre al punto de vista real. Si la imagen no cambiara a medida que nos desplazamos o giramos la cabeza, la sensación de inmersión se perdería.

Aunque no es imprescindible, lo normal es que la imagen que se presente sea estereoscópica (es decir, imágenes diferentes para cada ojo), permitiendo ver en tres dimensiones.

1.2.2. Sistemas proyectivos

En los sistemas proyectivos, se intenta proporcionar la misma sensación de inmersión al usuario, pero en lugar de utilizar un dispositivo acoplado a su cabeza, el usuario se introduce en una habitación o adminículo cerrado, en cuyas paredes se presentan o se proyectan una o más imágenes del mundo virtual. Este sistema está especialmente indicado para aplicaciones multiusuario, donde un grupo de personas comparte simultáneamente la experiencia.

El concepto es parecido al de las cápsulas de simulación tipo Venturer, en las que un grupo de personas pueden ver una película en el interior de la cápsula, mientras que ésta se inclina a un lado y a otro al compás de las imágenes, como si estuviera realmente desplazándose.

Lo único que diferencia a dichas cápsulas de un sistema de realidad virtual proyectivo es la interactividad. En las cápsulas tipo Venturer, lo que los usuarios pueden contemplar

es una película previamente grabada, y los movimientos de la cápsula también son predeterminados. En un sistema de realidad virtual proyectivo, por el contrario, es el usuario quien controla los desplazamientos por el mundo virtual, y las imágenes se calculan en tiempo real de acuerdo con los mismos.

Los simuladores de vuelo profesionales corresponden precisamente a este tipo de sistema de realidad virtual.

1.2.3. Sistemas de sobremesa

En los sistemas de sobremesa no se pretende proporcionar al usuario una sensación visual de inmersión en el mundo virtual. Las imágenes son presentadas en una pantalla de ordenador, por lo que el usuario no pierde la visión del mundo circundante.

Para conseguir el efecto de visión de relieve pueden emplearse dispositivos especiales como son las gafas estereoscópicas, aunque también existen sistemas de sobremesa monoscópicos, con imagen plana.

1.3. APLICACIONES

Tal vez sean las de ocio las aplicaciones más conocidas de la tecnología de realidad virtual. Sistemas como el Virtuality o el Cybertron han contribuido más que ninguna otra cosa a popularizar la realidad virtual en todo el mundo. Pero los campos de aplicación de la realidad virtual son múltiples.

Podemos clasificar las aplicaciones profesionales de la realidad virtual en cuatro diferentes categorías:

- Prototipado/modelado.
- Entrenamiento/simulación/enseñanza.
- Aplicaciones metafóricas.
- Teleoperación y tele-robótica.

En este apartado, vamos a pasar revista a los principales tipos de aplicaciones existentes, comenzando por las de carácter profesional. Es posible que algunos de los conceptos le resulten poco familiares, pero a lo largo del libro iremos analizando cada uno de ellos.

1.3.1. Prototipado

Las aplicaciones de prototipado y modelado son aquellas en que se emplea un sistema de realidad virtual para tratar de simular las condiciones reales de operación de un cierto equipo u objeto, con el fin de detectar cualquier problema antes de proceder a su construcción. Un ejemplo podría ser la Cocina Virtual de Matsushita, una aplicación que está en funcionamiento en Japón desde hace ya casi tres años, mediante la cual el futuro comprador puede diseñar interactivamente su cocina, colocando los muebles en distintas posiciones y observando el efecto final.

1.3.1.1. Diseño industrial

Los sistemas de realidad virtual están comenzando a ser usados en la industria automovilística, naval y aeronáutica, como ayuda para el diseño de nuevos modelos. Los fabricantes americanos de coches han establecido un consorcio para investigar las posibilidades de la RV, en el que participan varios organismos de investigación y el ejército americano, siendo Chrysler una de las empresas más activas en la investigación de las aplicaciones de este tipo de tecnología.

En el sector de la aviación, Boeing ha comenzado asimismo a investigar estas técnicas, aunque ha tropezado con la dificultad de que representar uno de sus nuevos modelos de avión en un sistema de RV requeriría una capacidad de generación de gráficos de entre 5.000 y 10.000 millones de polígonos por segundo, lo cual excede con mucho la capacidad de las estaciones más potentes del mercado.

En la industria naval, Lockheed y General Dynamics están investigando las posibilidades que la RV abre en el diseño de nuevos barcos y submarinos.

Quizá la empresa que con más éxito ha aplicado este tipo de técnicas hasta el momento, dentro de este sector de la industria, sea la empresa de fabricación de vehículos de construcción y obras públicas Caterpillar. Caterpillar está utilizando una aplicación de RV para comprobar las condiciones de visibilidad en el interior de los nuevos modelos de vehículos. La aplicación ha sido desarrollada para la empresa por el Centro Nacional de Aplicaciones de Supercomputación, de la Universidad de Illinois.

Se trata de un sistema inmersivo, con visiocasco, en el que el operador se sienta en un prototipo de cabina equipada con controles reales. Los controles reales son reproducidos en el mundo virtual, de forma que el operador puede verlos. El operador puede mover el vehículo virtual y accionar los controles de los distintos elementos, como la pala de una excavadora, etc... El casco le permite dirigir su vista en todas las direcciones del mundo virtual.

El objetivo es comprobar si, en condiciones reales de operación, el usuario del vehículo tiene la adecuada visibilidad en todas direcciones. Antes, este tipo de comprobaciones no podían realizarse sin construir un costoso prototipo del vehículo completo. Si se descubría un problema, el prototipo tenía que ser rechazado.

Gracias a este sistema, Caterpillar ha podido acortar en 9 meses el tiempo de diseño de sus nuevos modelos, que saldrán al mercado en 1996. Asimismo, se ha eliminado el coste de los prototipos fallidos.

El sistema emplea cuatro ordenadores diferentes: una estación Silicon Graphics para la generación de los gráficos, una estación HP para la simulación dinámica, un PC para controlar los elementos de la cabina, y un Macintosh para la generación de sonido. El software de desarrollo de la aplicación de RV fue el WorldToolKit, de Sense8.

1.3.1.2. Diseño arquitectónico

Las posibilidades de prototipado que los sistemas de RV ofrecen presentan numerosas aplicaciones en arquitectura, urbanismo y decoración. Los sistemas de realidad virtual permiten disponer del prototipo de un edificio o de una habitación antes de proceder a su construcción o decoración. A esta categoría pertenecen la aplicación de la Cocina Virtual de Matsushita, antes mencionada, o el programa para decoración de interiores desarrollado por el Instituto de Gráficos por Ordenador de Fraunhofer (Alemania), denominado Virtual Design.

Utilizando un sistema de realidad virtual, el LISSIT de la Universidad Politécnica de Valencia diseñó la Ciudad de la Ciencia de Valencia.

Otra aplicación algo más curiosa es la desarrollada por el Centro Hines de Rehabilitación, en Illinois, en la que se utilizaba un sistema de RV al que se había acoplado una silla de ruedas, con vistas a la adecuación de una nueva casa para las personas impedidas.

1.3.1.3. Planificación de operaciones

La posibilidad de construir réplicas virtuales de los objetos, permite realizar simulacros de operaciones complejas, con vistas a detectar posibles fallos antes de llevar a cabo la acción real. Éste fue el caso, por ejemplo, de la reparación del telescopio espacial Hubble. Dicho telescopio, que había sido puesto en órbita en abril de 1990, tenía un problema de aberración esférica en el espejo principal que disminuía su capacidad de captación de la luz en un 80%. El problema sólo fue detectado una vez que el telescopio se encontraba ya en órbita.

En consecuencia, la NASA decidió enviar una misión en la lanzadera espacial para corregir el problema del telescopio mediante el montaje de un juego de espejos. Para detectar con la mayor antelación posible los problemas de montaje que pudieran surgir, se decidió simular la reparación mediante un sistema de realidad virtual.

El sistema de RV fue diseñado por Lockheed. Era un sistema de sobremesa, que empleaba una estación de Silicon Graphics y gafas estereoscópicas Crystal Eyes, de StereoGraphics. El sistema incluía la simulación de los elementos ópticos, para determinar con exactitud el recorrido de los haces de luz entre los espejos, de acuerdo a la posición de éstos.

La simulación llevada a cabo permitió verificar la adecuación de las distintas piezas mecánicas, corregir los defectos de las mismas, y comprobar que la situación final del juego de espejos era la adecuada para corregir la "miopía" del Hubble.

1.3.1.4. Paseos virtuales

También resulta posible emplear sistemas de realidad virtual con el fin de reproducir escenarios lejanos, desaparecidos o inaccesibles. Con un sistema de realidad virtual, igual podemos explorar un monumento chino que visitar una ciudad de la antigua Roma.

El sistema de Exploración Planetaria Virtual (Virtual Planetary Exploration), desarrollado en el NASA Ames Center, permite llevar a cabo paseos virtuales en un paisaje

construido a partir de datos reales de Marte, extraídos de las imágenes enviadas por el Viking. El proyecto fue iniciado por Michael McGreevy en 1988, finalizándose el primer prototipo a mediados de 1990. El sistema puede emplearse de modo inmersivo (utilizando un visiocasco) o no inmersivo. El único problema con el modo inmersivo es que, mientras tiene puesto el visiocasco, el usuario no puede utilizar el teclado, que se requiere para determinadas operaciones. Asimismo, no se puede proporcionar información textual al usuario, por la baja resolución que los visiocascos proporcionan. Por eso, para determinadas actividades, es preferible utilizar el modo de trabajo no inmersivo. Como dispositivos de entrada, además del teclado y el ratón, se utilizan un Joystick 3D y un electroguante.

1.3.2. Simulación

Las aplicaciones de simulación fueron las que dieron lugar, en buena medida, al nacimiento de los sistemas de RV. Las aplicaciones de simulación se utilizan para entrenar a los operadores humanos en el manejo de determinados equipos o en la realización de ciertas actividades.

El entrenamiento mediante técnicas de simulación tiene sentido cuando el entrenamiento en condiciones reales es peligroso o muy costoso, como sucede con los simuladores de vuelo, o cuando no resulta posible realizarlo en condiciones reales, como es el caso de una misión de exploración planetaria. Pero las técnicas de simulación presentan ventajas adicionales, como son el hecho de que se dispone de un perfecto control de las condiciones de entrenamiento. Así, un simulador de vuelo permite recrear cualquier condición que deseemos, en cualquier momento: vuelo nocturno, niebla, averías en la aeronave.

Además de los simuladores de vuelos, podemos citar las redes de simulación de batallas terrestres desarrolladas por el Pentágono, como la red SIMNET o la NPSNET; las aplicaciones de entrenamiento de bomberos; los simuladores de conducción, como el simulador construido por el LISSIT en Valencia; o el Sistema de entrenamiento extravehicular para astronautas, desarrollado en Holanda por la Netherlands Organization for Scientific Research.

En general, los sistemas de realidad virtual encuentran aplicación en todos aquellos campos donde el disponer de un sistema simulado permite obtener alguna ventaja (económica, de seguridad o de cualquier otro tipo) sobre la utilización de un entorno real. Así, Georgia Tech está diseñando un sistema de RV para rehabilitación que consiste en un ascensor virtual de cristal que tiene una altura de 50 plantas. Con este sistema, resulta posible tratar en condiciones simuladas a los individuos que padecen acrofobia.

En el campo de la enseñanza, el Johnson Space Center de la NASA ha desarrollado un laboratorio virtual de física, en el que los alumnos pueden utilizar un panel de control virtual flotante para regular parámetros tales como la masa, la gravedad, la fricción o el tiempo, y observar el efecto que ello produce sobre los objetos.

También en el mercado de consumo encuentran su aplicación los sistemas de simulación. Como nota curiosa, la empresa americana Tectrix ha desarrollado un simulador ciclista, el VR Bike. Dotado con un monitor de 20 pulgadas y altavoces estereofónicos, va presentando

al usuario un paisaje simulado a medida que éste pedalea. Para dar mayor sensación de realismo, la resistencia de los pedales varía según el usuario esté bajando o subiendo una cuesta, y la bicicleta se inclina ligeramente a uno y otro lado en las curvas. Incluso dispone de un ventilador para proporcionar la sensación del viento en la cara.

1.3.3. Aplicaciones metafóricas

En las aplicaciones metafóricas, lo que se pretende es usar un sistema de RV para dar una representación visual a un cierto conjunto de datos. En esta categoría caen todas las aplicaciones de visualización científica de carácter inmersivo.

En la Universidad de Carolina del Norte se ha desarrollado, por ejemplo, una aplicación para el diseño de nuevos fármacos en la que se utiliza un modelo molecular tridimensional, y dispositivos de realimentación cinestésica para controlar las reacciones químicas. También en la Universidad de Carolina del Norte se ha desarrollado otra aplicación, denominada el Nanomanipulador, que es una interfaz virtual para el control de un microscopio de efecto túnel, en la que se utilizan, asimismo, un visiocasco y un dispositivo de realimentación cinestésica. Con esta herramienta se puede ver una representación virtual de los datos procedentes del microscopio, y modificar la superficie del material utilizando pulsos de tensión. Es una especie de sistema de telepresencia, pero a escala atómica.

Otro sistema de RV, desarrollado en la Universidad de Illinois, es un programa de exploración de fractales, que permite interaccionar con una imagen fractal, cambiando sus atractores, los modos de visualización, etc...

Básicamente, lo que se hace en una aplicación metafórica es aprovechar la imagen para aumentar nuestra capacidad de comprensión de un determinado conjunto de datos. En otras palabras, una aplicación metafórica es una ayuda a la toma de decisión.

El túnel del viento virtual, por ejemplo, desarrollado en 1991 por Steve Bryson y Creon Levit en el NASA Ames Center, es un sistema inmersivo que permite visualizar desde cualquier punto, en tres dimensiones, el flujo de aire alrededor de una aeronave. Ayudándose de un electroguante, el científico puede situar y mover a voluntad diversos tipos de puntos de control en el flujo de aire, para ver las características de éste (vectores de velocidad y su evolución con el tiempo, trayectoria de partículas dentro del flujo, etc...). Se trata de una aplicación en la que priman los cálculos de fluidodinámica, que consumen la mayor parte de los recursos de procesamiento y memoria. En 1992 Bryson y Yamasaki desarrollaron la versión distribuida del sistema, con la que diversos científicos pueden trabajar conjuntamente sobre el mismo diseño.

Pero las aplicaciones metafóricas no se limitan al campo científico. Un ejemplo es el programa Metaphor Mixer, para el manejo de carteras de valores de bolsa, desarrollado por la empresa Maxus Systems International de Nueva York, y que lleva funcionando desde 1992. Con el Metaphor Mixer, los inversores utilizan una interfaz de realidad virtual para visualizar en tiempo real las cotizaciones de los distintos valores, y gestionar sus acciones. Otro ejemplo serían las investigaciones que British Telecom lleva actualmente a cabo para aplicar las tecnologías de RV al análisis de tráfico de telecomunicaciones.

1.3.4. Tele-robótica

Los sistemas de teleoperación y tele-robótica son sistemas donde se emplean técnicas de RV para que un operador pueda controlar a distancia la acción de un robot o dispositivo mecánico. Las imágenes que se le presentan al operador en este tipo de sistemas suelen ser una mezcla de imágenes reales y sintéticas, ya que se usan metáforas gráficas para representar diversos parámetros de interés. Por ejemplo, en un sistema de exploración oceanográfica como el que está siendo desarrollado por la empresa IBM, la ausencia de luz en las profundidades marinas impide al operador del vehículo telecontrolado disponer de una visión de lo que está haciendo. Para suplir esta carencia, se obtienen datos del entorno del vehículo utilizando técnicas de telemetría y, a partir de dichos datos, se genera una representación sintética de la escena.

Es este hecho de utilizar las técnicas de RV para complementar los datos reales lo que hace que, al hablar de este tipo de aplicaciones, se utilice con frecuencia el término de "realidad aumentada" (augmented reality).

Además del entrenamiento del operador utilizando técnicas de simulación y de las aplicaciones metafóricas, los sistemas de RV tienen tres aplicaciones fundamentales en el campo de la teleoperación y la tele-robótica:

- Programación del robot mediante un robot virtual. El manejo de los robots o equipos mecánicos puede requerir complejos instrumentos de control, que pueden ser bastante difíciles de manejar por el operador humano. Una de las posibilidades, de cara a evitar errores de operación, consiste en emplear una réplica virtual del robot, y hacer que dicha réplica ejecute las órdenes que el operador va emitiendo. Puesto que se trata de una simulación, el operador puede detenerse cuando lo desee, volver atrás y repetir una cierta operación hasta quedar satisfecho del resultado. Una vez que los movimientos del robot virtual son los deseados, los comandos del operador que han sido grabados se envían de un golpe al robot real, quien ejecuta los movimientos en secuencia.
- Situaciones donde existe un gran retardo de transmisión. En el caso, por ejemplo, de estar telecontrolando un vehículo situado en la superficie de Marte, los retardos de transmisión impiden ver de forma inmediata la consecuencia de las acciones que el operador ejecuta. Para poder obviar este problema, puede emplearse una réplica virtual del vehículo y de la escena por la que éste se mueve. Cuando el operador da una orden de movimiento, ve el resultado de la acción en el vehículo virtual. Simultáneamente, la orden se transmite al vehículo real. El sistema comprueba de manera constante que la situación del vehículo virtual en la escena virtual se corresponde con la que el vehículo real tiene en la escena real, y ajusta la posición del vehículo virtual de acuerdo a las discrepancias detectadas.
- Situaciones de falta de luz, como es el caso de la exploración oceanográfica, antes mencionado. En estas circunstancias, lo que se hace es utilizar la síntesis de imágenes para reconstruir la escena a partir de datos no visuales.

En estos momentos, los problemas actuales con los que se enfrenta la investigación en el campo de la tele-robótica provienen de la dificultad de simular los aspectos dinámicos (no

meramente geométricos) de los objetos: deformaciones, inercia, rozamientos, ... Por ejemplo, el simular una pared virtual y un robot virtual desde el punto de vista meramente geométrico es sencillo, de forma que el operador puede ver en todo momento la situación del robot virtual con respecto a la pared. El simular el contacto con la pared, cuando el brazo del robot la alcanza, también es simple, empleando mandos de control con realimentación cinestésica. Pero si queremos una simulación física más perfecta, por ejemplo, la deformación de la pared, las técnicas actuales de simulación se quedan cortas. Y el problema es que, en muchas operaciones de tele-robótica, es necesario conseguir simular estos aspectos si se quiere que la efectividad de los operadores sea la adecuada.

1.3.5. Aplicaciones de ocio

1.3.5.1. Juegos de galería

Hace ya años que existen, en las galerías de videojuegos, simuladores de vuelo y de conducción cuyas características rozan la frontera de la realidad virtual. Hasta hace relativamente poco tiempo, las limitaciones de carácter técnico obligaban a recurrir a tretas bastante sofisticadas para dar a este tipo de juegos una adecuada sensación de realismo. El año 1993, sin embargo, marcó un hito al registrar la aparición de los primeros videojuegos que hacían un uso extensivo de las técnicas de generación de gráficos 3D. Pero volvamos la vista hacia atrás por unos momentos.

Hasta 1989, la creación de los gráficos para los juegos se realizaba por un procedimiento similar al empleado para la confección de películas de dibujos animados, utilizando sprites (objetos animados capaces de desplazarse por la pantalla) representativos de los distintos actores y objetos incluidos en el juego.

Con una adecuada planificación de los elementos del juego, se pueden conseguir efectos espectaculares utilizando unos recursos (en términos de memoria y capacidad de proceso) relativamente limitados. Sin embargo, esta técnica tiene sus desventajas, la más importante es que sólo proporciona una visión plana del juego, en la que el usuario ve las imágenes "desde fuera".

Por este motivo las técnicas de animación basadas en sprites no son adecuadas para el desarrollo de juegos de simulación (carreras de coches, simuladores de vuelo), donde es imprescindible que la acción transcurra en primera persona. Es decir, es imprescindible que la cámara se desplace siguiendo los movimientos del vehículo.

Las tecnologías necesarias para la obtención de simuladores realistas hace mucho tiempo que estaban disponibles (eso si, a un coste muy elevado), a partir de los sistemas de simulación desarrollados para el mercado militar. En este tipo de sistemas, una serie de elementos hardware y software especializados se utilizaban para generar objetos complejos en tiempo real a partir de su descomposición en polígonos, y de acuerdo con los desplazamientos del usuario a través del mundo virtual. En el mismo año 1989, la compañía americana de videojuegos Atari lanzó por vez primera al mercado un producto, denominado Hard Drivin', que empleaba las técnicas de generación de gráficos propias de los simuladores de carácter militar, con lo que se conseguía un realismo mucho mejor que el de cualquier otro videojuego

hecho hasta la fecha. La pantalla esencialmente plana y bidimensional de los videojuegos anteriores era sustituida en Hard Drivin' por un mundo tridimensional que eliminaba las restricciones a la conducción.

La compañía japonesa Namco, que había tenido anteriormente múltiples contactos con Atari, fue la primera en reaccionar y sacar al mercado un producto, Winning Run, que empleaba técnicas de generación de gráficos poligonales. Posteriormente sacaría al mercado Cyber Sled, un juego de combate entre vehículos robot[1] que alcanzaba los 60.000 polígonos por segundo.

La siguiente compañía en desarrollar sus propios módulos de tratamiento de gráficos 3D fue Sega, que comenzó en 1993 a sacar al mercado los primeros videojuegos que, por su capacidad de tratamiento de gráficos 3D, podemos considerar como auténticos sistemas de realidad virtual. El primero de dichos productos fue Virtua Racing, un juego de carreras de coches en el que el usuario podía elegir entre cuatro diferentes posiciones de la cámara, y en el que se generaban los gráficos correspondientes a los vehículos y al paisaje a una velocidad de 180.000 polígonos/segundo.

El siguiente producto de Sega con estas características que salió al mercado fue Virtua Fighting, un juego de combate entre dos luchadores que empleaba el mismo soporte hardware que Virtua Racing y que ha alcanzado un considerable éxito. Virtua Fighting, sin embargo, permitió extraer dos importantes lecciones.

La primera fue que la tecnología de tratamiento de objetos 3D no era aún lo suficientemente avanzada como para permitir un adecuado realismo en los juegos de combate. Estos juegos emplean lo que se denomina "interacciones de corta distancia": cada golpe que un actor proporciona a otro obliga a una respuesta gráfica muy precisa (se debe acusar el golpe en el momento adecuado y en la dirección correcta, se debe lanzar una secuencia de animación apropiada que muestre cómo se encaja el golpe, etc...). Los juegos de carreras, por el contrario sólo presentan "interacciones de larga distancia", en las que los objetos no interactúan entre sí de forma que obliguen a un tratamiento gráfico especial (los coches no se deforman al rozarse y, en el caso más extremo, que es cuando se produce un choque, basta con lanzar la secuencia de destrucción del coche, sin excesiva precisión en cuanto al tiempo o la dirección).

Debido a ello, el combate en Virtua Fighter no tiene lugar propiamente en tres dimensiones. Aunque las figuras son tridimensionales, sus movimientos se realizan en un único plano. Para enmascarar esta restricción, y también para aumentar la sensación de tridimensionalidad, la cámara está perpetuamente bailando alrededor de los dos contendientes.

La segunda lección fue que la generación de objetos poligonales, por sí sola, distaba mucho de conseguir las dosis de realismo gráfico de los juegos realizados con sprites. De

1 En determinados países, como Alemania, los juegos de combate entre personajes con aspecto humano están sujetos a una serie de restricciones de comercialización, por su carácter violento. Pero si el combate tiene lugar entre robots la cosa cambia, y no hay restricciones.

hecho, los actores de Virtua Fighting son bastante menos realistas que los que se pueden encontrar en anteriores juegos bidimensionales como Super Street Fighter o Mortal Kombat.

Mientras tanto, la tecnología de simulación continuaba su marcha, y ya habían empezado a hacer su aparición las técnicas de texturado, que permitían obtener un realismo extraordinario mediante el procedimiento de proyectar una imagen artificial o digitalizada sobre las superficies poligonales de los objetos.

Dichas técnicas de texturado dejaban completamente obsoletos los anteriores productos, por lo que las compañías japonesas se vieron obligadas a desarrollar nuevos elementos hardware que incorporaran las nuevas tecnologías, para lo cual optaron por llegar a acuerdos con compañías ya introducidas en el sector de la simulación y la generación de gráficos 3D. Namco optó por asociarse con Evans & Sutherland, mientras que Sega acudió a la compañía Martin Marietta para la incorporación de técnicas de texturado en sus nuevos módulos de generación de imágenes.

De nuevo se adelantó Namco a Sega, al sacar al mercado un juego de carreras, Ridge Racer, en el que las técnicas de texturado, unidas a una capacidad de generación de 240.000 polígonos por segundo, proporcionan al paisaje un realismo verdaderamente increíble. Este realismo, unido a unos controles muy bien diseñados y a que el juego estaba muy bien cuidado desde el punto de vista de la "jugabilidad", hizo de Ridge Racer un auténtico éxito de ventas.

Pero, aunque con algún retraso, la respuesta de Sega no desmerece en absoluto con respecto a Namco. El nuevo juego de carreras de Sega, Daytona, sacado al mercado a mediados de 1994, presenta una calidad de imagen soberbia, gracias a los nuevos módulos hardware desarrollados por la compañía japonesa.

Mientras tanto, Nintendo prepara su retorno al mercado de las galerías de videojuego, para lo cual ha llegado a un acuerdo con Silicon Graphics para el desarrollo de sus propios módulos de generación de gráficos 3D.

1.3.5.2. Juegos inmersivos

Más sofisticados son los juegos inmersivos. Entre los basados en visiocasco destacan los sistemas Virtuality, que se empezaron a comercializar en 1990. El software de las máquinas Virtuality ha sido desarrollado por la compañía californiana CyberStudio, que fue creada por acuerdo de W Industries, Horizon Entertainment y Spectrum Holobyte. Los títulos son Legend Quest, ExoRex, Dactyl Nightmare, Zone Hunter o Virtuality Boxing.

La serie Virtuality 1000, de la que se han vendido centenares de equipos, consta de dos modelos diferentes. En el Virtuality 1000 CS, el jugador está de pie, equipado con un visiocasco y un mando de control con el que realiza las acciones correspondientes al juego. En el modelo Virtuality 1000 SD, el jugador está sentado, y los periféricos de los que el usuario dispone son los mismos, excepto que puede tener un volante en lugar de un mando de control.

Los equipos Virtuality son monousuario, aunque ya en enero de 1992 Virtuality decidió expandir su campo de aplicación, abriendo en Nottingham un centro denominado Legend Quest, con cuatro máquinas Virtuality 1000CS conectadas en red. En Legend Quest, los

jugadores participan en un juego de rol, luchando juntos contra los monstruos que les salen al paso, para derrotar al amo del laberinto.

Antes de empezar, los jugadores pueden escoger la apariencia gráfica que van a tener en el juego, eligiendo el tipo de héroe que desean ser (mago, guerrero o ladrón), la raza (elfo, humano o enano) y el sexo, así como ciertas características como la altura o el color del pelo. El sistema dispone de un equipo de procesamiento de la voz, de forma que la voz del usuario es modulada para ajustarse a las características de su personaje.

Los jugadores pueden interrumpir la partida en el momento que lo deseen, grabándose el estado del sistema en una llave magnética. De esa forma, pueden reemprender la aventura cualquier otro día en el punto exacto donde la dejaron.

Otro equipo, del que recientemente se ha hecho una instalación en nuestro país, es el Cybertron, de la empresa americana Cyber Event Group, en el que el jugador se amarra a un giroscopio de tres ejes, con el que puede rotar en todas las direcciones bajo comando del sistema y de acuerdo con las incidencias del juego.

1.3.5.3. Juegos de plataforma

Los juegos de plataforma representan el segmento más alto (y caro) de los juegos de realidad virtual. En los juegos de plataforma, la sensación de inmersión se consigue no obligando al usuario a aislarse del mundo exterior mediante un visiocasco, sino introduciéndolo en una cápsula cerrada, en la que el mundo virtual se muestra mediante una o más pantallas de proyección de tamaño suficiente, como si fueran las ventanillas de una nave a través de las que el usuario contempla el mundo exterior.

La imagen que el usuario vea a través de las "ventanillas" vendrá determinada por la posición de la nave dentro del mundo virtual. Este tipo de cápsula se monta sobre una plataforma hidráulica o neumática controlada por un ordenador, de forma que se pueden transmitir al usuario las sensaciones adecuadas de movimiento y giro, simulando la inercia y la aceleración mediante inclinaciones de la plataforma.

Este tipo de cápsula resulta bastante familiar hoy en día. En España hay numerosas cápsulas del tipo Venturer o Atlas, en las que hasta sesenta usuarios pueden compartir la experiencia de un viaje simulado por la nieve o por el espacio. Estas cápsulas de simulación no constituyen en sí mismas sistemas de realidad virtual, dado que carecen de una característica fundamental de dichos sistemas: la interactividad. En ellas, los usuarios experimentan un conjunto de movimientos que les hacen la ilusión de participar en un viaje. Pero dicho viaje está previamente programado, es fijo: el usuario no puede hacer nada para controlar su experiencia, estando limitado a sentirla.

Entre las empresas pioneras en el campo de los simuladores interactivos hay que citar a Virtual World Entertainment, empresa creadora de los Battle Tech Centers. En dichos centros, similares a un parque temático, los usuarios luchan entre sí en tiempo real a los mandos de un macro-robot. El primero de los centros fue inaugurado en 1990, y VWE ha abierto hasta la fecha quince centros en diferentes países. Ante el éxito de este tipo de centro de atracciones, otras empresas, como VOR, se han lanzado al mercado.

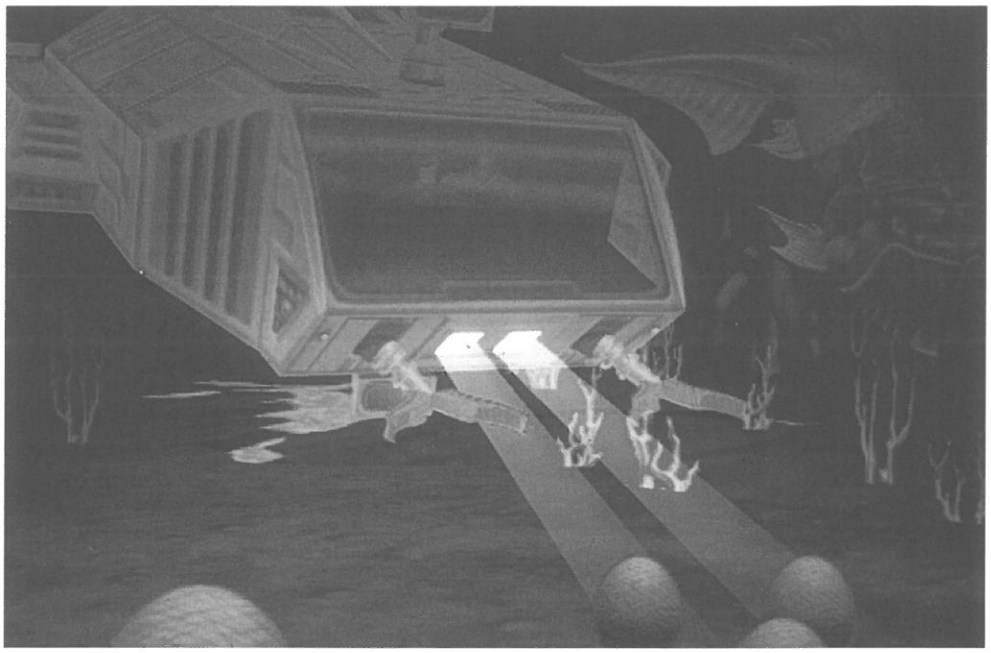

Figura 1.4. Viaje submarino de Iwerks Entertainment.
(Fotografía de Evans & Sutherland)

Diversas empresas fabrican otros tipos de simuladores. Así, Hughes Aircraft está desarrollando un simulador de viajes espaciales, para su instalación en centros temáticos, mientras que Paramount Pictures y Spectrum Holobyte comenzaron a finales de 1992 el desarrollo de un simulador basado en la serie Star Trek, con el que tienen previsto abrir cincuenta centros de ocio. Evans & Sutherland e Iwerks Entertainment también han empezado a comercializar en el segundo semestre de 1994 unos simuladores de viaje submarino, y la empresa inglesa Rediffusion Simulation, especializada en la construcción de simuladores no interactivos, ha anunciado el Commander, su primer sistema interactivo de simulación.

La frontera que separa los juegos de plataforma de las máquinas para galería de videojuego es muy estrecha. Las máquinas de galería más caras incorporan cabinas hechas a medida, en las que el usuario se introduce para manejar unos controles que remedan con más o menos fidelidad los de la cabina de un coche o un avión. Por supuesto, dichas cabinas no incorporan los costosos mecanismos hidráulicos de los juegos de plataforma, y suelen ser abiertas (no requiriendo el desarrollo de una cápsula), pero en los próximos años se vivirá una rápida convergencia de estos dos mercados. En la gama baja, los juegos de plataforma monousuario invadirán las galerías de videojuegos a medida que los costes decrezcan lo suficiente como para que la inversión en este tipo de máquinas sea rentable. En el extremo alto del mercado, los simuladores multiusuario más sofisticados y costosos se instalarán en parques temáticos o en parques dedicados, al estilo de Battle Tech.

2
Panorámica general

2.1. SISTEMAS DE RV: QUÉ SON

2.1.1. Definición

Vamos a comenzar nuestra visita introductoria al mundo de la realidad virtual tratando de entrever cuáles son sus límites, es decir, qué entendemos por sistema de realidad virtual.

Como vamos a ver, es éste un concepto hasta cierto punto difuso. La realidad virtual abarca, hoy en día, distintos tipos de sistemas, aplicaciones y tecnologías, de forma que la tarea de definirlo de manera precisa es punto menos que imposible.

No es que la definición no exista. De hecho, podemos definir un sistema de realidad virtual como

> **un sistema interactivo que permite sintetizar un mundo tridimensional ficticio, creando en el usuario una ilusión de realidad.**

El problema es que esta definición no es, en modo alguno, precisa, porque descansa en un factor psicológico completamente subjetivo: la capacidad de crear en el usuario una ilusión de realidad.

Sin embargo, es fácil poner ejemplos de sistemas que se ajustan a dicha definición. El sistema hipotético que presentamos en el capítulo anterior es un ejemplo claro de sistema interactivo capaz de sintetizar un mundo virtual y de crear en el usuario la ilusión de que se encuentra, efectivamente, inmerso en dicho mundo.

Otro ejemplo, esta vez real: el de los simuladores de vuelo profesionales. Un simulador de vuelo es un sistema utilizado para entrenar a los pilotos en la conducción de aeronaves. Los simuladores de vuelo profesionales son equipos tremendamente costosos y sofisticados, en los que cada elemento está concebido para que el entrenamiento sea lo más parecido posible a la conducción real de un avión: desde la propia mecánica del simulador, igual a la de la aeronave que se pretende remedar, hasta los mandos que el piloto utiliza, pasando por el más importante de los elementos: la información de carácter visual.

En un simulador de vuelo, una serie de potentes equipos de procesamiento y de elaborados programas informáticos permiten recrear el aspecto del mundo exterior a la

carlinga del avión, *como si* el piloto se encontrara realmente volando sobre un determinado terreno. Cuando el piloto efectúa una maniobra, la imagen proyectada por el simulador se desplaza, tal y como lo haría la imagen real en caso de pilotar un avión auténtico, y la carlinga donde el piloto se encuentra situado se mueve de manera similar a como la de un avión real lo haría.

De hecho, para que la simulación sea lo más perfecta posible, pueden utilizarse datos reales para construir la imagen, de forma que ésta sea una reproducción más o menos fiel del terreno de un cierto país (idealmente, del país sobre el que luego va a volar el piloto).

Está claro que este sistema responde, asimismo, a nuestra definición de realidad virtual. Un simulador de vuelo recrea un mundo de tres dimensiones por el que el usuario puede desplazarse a voluntad. En cuanto a la ilusión de realidad, hasta tal punto es ésta intensa que no son infrecuentes los ataques de nervios entre aspirantes a piloto después de un choque simulado, causado por una maniobra equivocada.

2.2. CARACTERÍSTICAS DE LOS SISTEMAS DE RV

La definición de sistema de realidad virtual que hemos dado engloba cuatro diferentes conceptos:

- Capacidad sintética.
- Interactividad.
- Tridimensionalidad.
- Ilusión de realidad.

En primer lugar vamos a pasar revista a todos ellos, con el fin de precisar algo más la definición, lo cual nos ayudará a identificar los mecanismos básicos mediante los cuales los sistemas de RV cumplen su objetivo.

2.2.1. Capacidad sintética

En un simulador de vuelo, las imágenes que en cada momento se presentan al usuario no están almacenadas en ninguna parte. Son imágenes sintéticas, generadas en tiempo real por el ordenador, de acuerdo a la posición que en ese momento ocupa el usuario dentro del mundo virtual.

Puesto que el usuario tiene una libertad casi ilimitada de movimiento, pudiendo situarse en cualquier punto que desee del mundo virtual, resultaría imposible calcular y almacenar de antemano las imágenes correspondientes a todas y cada una de las posibles posiciones.

En lugar de ello, el ordenador mantiene una base de datos donde consta la forma de cada uno de los objetos y la posición que ocupa. Simplificando mucho, podríamos decir que, a la hora de generar la imagen, el ordenador recorre la lista de objetos, calcula cómo se vería cada uno desde la posición que el usuario ocupa, teniendo en cuenta la distancia, y lo dibuja en la correspondiente posición de la pantalla.

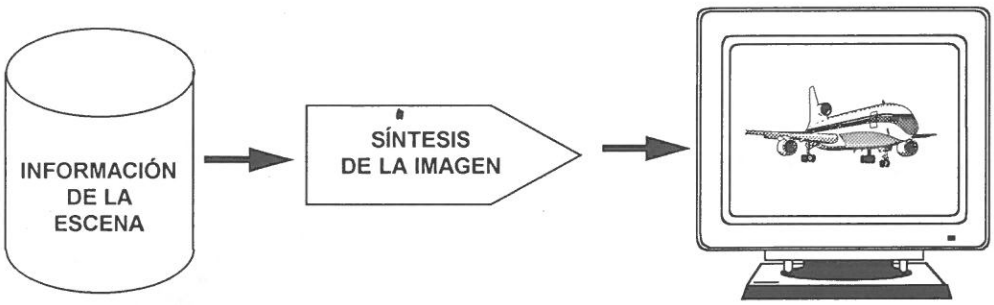

Figura 2.1. Síntesis de una escena

Es decir, el mundo virtual no es un todo compacto, sino que está descompuesto en una serie de objetos. Las imágenes se sintetizan a partir de las correspondientes a cada uno de esos componentes.

Obsérvese la diferencia que hay entre **generar** y **reproducir** las imágenes. Existen en la actualidad numerosos sistemas que permiten ver imágenes de video en tres dimensiones, utilizando pantallas o gafas especiales. Pero eso no son sistemas de realidad virtual, porque las imágenes de video han sido previamente grabadas, y no se sintetizan en tiempo real.

Es la misma diferencia que existe entre un tocadiscos y un sintetizador de música. Un tocadiscos permite *reproducir* secuencias musicales previamente grabadas. Podemos reproducirlas con mayor o menor volumen, podemos dar mayor volumen a unas frecuencias que a otras, pero no podemos alterar el contenido de la secuencia, que está ya fijado de antemano. Un sintetizador, por el contrario, *genera* las secuencias musicales a partir de una representación abstracta. Nosotros podemos, en cualquier momento, cambiar esa representación abstracta y generar una secuencia diferente. Un tocadiscos permite reproducir sólo aquellas secuencias que se graben previamente; un sintetizador permite generar un número infinito de secuencias distintas.

Eso mismo es lo que hace un sistema de realidad virtual. Los sistemas de RV contienen una representación abstracta del mundo virtual: cuáles son los objetos que lo componen, dónde están situados, cuáles son sus representaciones gráficas o sonoras, etc... a partir de la cual generan la información que luego presentarán al usuario.

2.2.2. Interactividad

Como hemos visto, los simuladores de vuelo (y todos los sistemas de realidad virtual) son capaces de sintetizar las imágenes en tiempo real, de acuerdo con las posiciones que ocupan los objetos y el propio usuario.

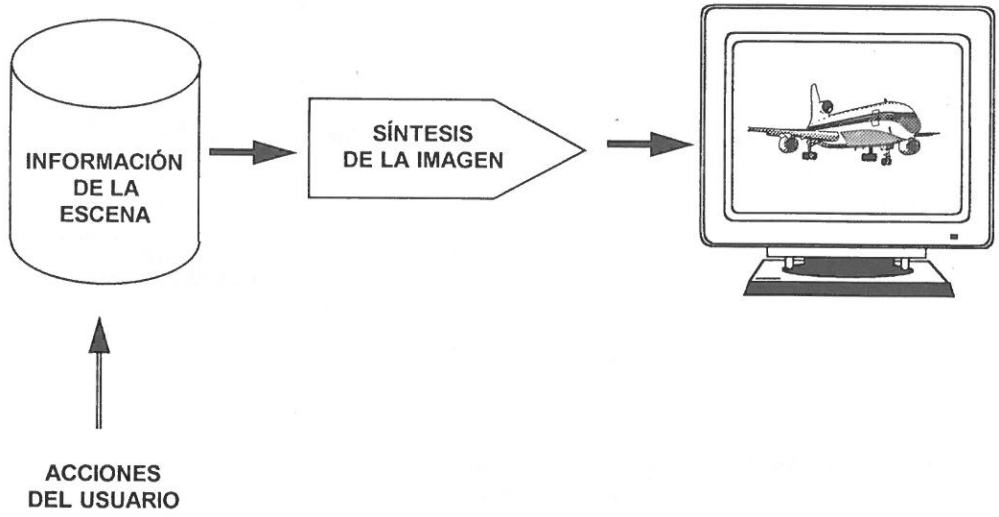

Figura 2.2. Aplicaciones interactivas

Pero esa capacidad sólo tiene sentido precisamente para permitir al usuario interaccionar con el sistema. Si el usuario no tuviera la posibilidad de influir sobre el estado del mundo virtual de ninguna forma, o cambiar su punto de vista de la escena, entonces no sería necesario sintetizar las imágenes, porque bastaría con grabar de antemano la película de lo que el usuario va a ver.

Es el caso, por ejemplo, de los simuladores tipo Venturer, que son cápsulas montadas sobre plataformas hidráulicas, en las que puede introducirse un grupo de personas y compartir la experiencia simulada de un descenso por la nieve o un viaje en una montaña rusa. En este tipo de cápsulas, el usuario contempla las imágenes de una película de video previamente grabada, mientras que la plataforma se mueve al compás de lo que la pantalla muestra. Los usuarios no tienen ningún tipo de control sobre la experiencia, sino que ésta discurre por un cauce prefijado de antemano.

En un sistema de RV, por el contrario, el usuario dispone de una mayor o menor dosis de control, pudiendo influir sobre lo que el sistema va a mostrarle.

Obsérvese cómo esta característica está relacionada con la anterior: si no existiera interactividad, no sería necesaria la capacidad de sintetizar las imágenes.

Existen dos tipos distintos de interacción entre el sistema y el usuario. Normalmente, los objetos que compongan el mundo virtual tendrán un cierto comportamiento asociado, y las acciones que esos objetos realicen se verán influidas por las del usuario. Así, es posible que en una aplicación de realidad virtual el usuario pueda desplazar un obstáculo, poner en

marcha un mecanismo, efectuar un disparo, o cualquier otra acción que haga que los objetos desaparezcan, cambien su representación gráfica o varíen su posición. Esto es lo que podríamos denominar **interacción dinámica,** y casi todos los sistemas de RV proporcionan este tipo de interacción de una u otra forma.

Pero, aun cuando un sistema no proporcione al usuario ninguna posibilidad de influir sobre el estado del mundo virtual, al menos debe permitirle otro tipo más básico de interacción, que consiste en que el usuario pueda cambiar su punto de vista dentro de la escena. A esto es a lo que llamamos **navegación** del mundo virtual.

En este tipo de aplicación, el usuario sería un mero espectador de la escena, pero no completamente pasivo: puede desplazarse por la misma para contemplarla desde otro ángulo, o desde otra posición. Este tipo de aplicación, donde el usuario no tiene a su alcance ninguna otra forma de interacción, aparte de las facilidades de navegación, recibe el nombre de **paseo virtual**.

Un parámetro fundamental de los sistemas interactivos es el de la velocidad de respuesta, que en los sistemas de RV se denomina **latencia**. La latencia es el tiempo transcurrido entre el instante en que el usuario efectúa una determinada acción, y el momento en que el sistema actualiza la información de salida (especialmente la de tipo gráfico) que entrega al usuario. En un simulador de vuelo, por ejemplo, la latencia sería el tiempo medido desde que el piloto efectúa una determinada maniobra hasta que el sistema muestra la nueva imagen en pantalla.

La latencia tiene una considerable importancia, porque en algunos sistemas puede llegar a tener una magnitud suficiente como para causar molestias físicas a los usuarios (ver el apartado dedicado a la latencia, en este mismo capítulo). Normalmente, la latencia es más problemática en los sistemas que incorporan algún dispositivo para medir la posición u orientación del usuario, porque la mayoría de dichos dispositivos son bastante lentos.

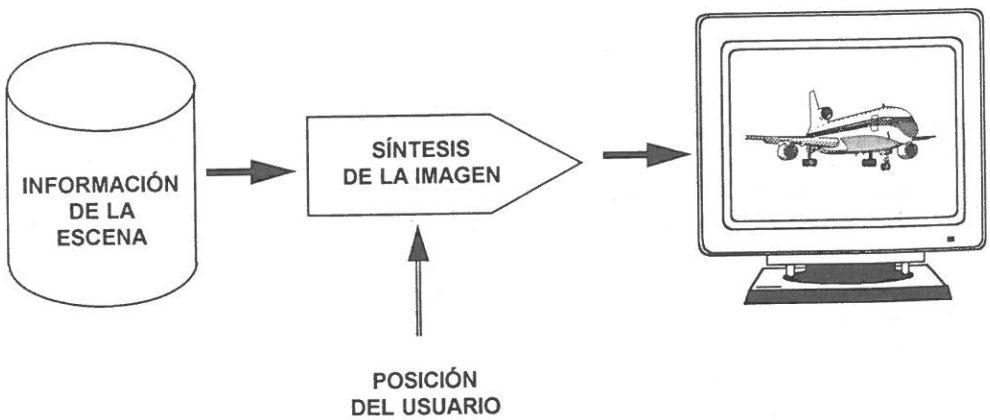

Figura 2.3. Aplicación de paseo virtual

En un sistema inmersivo, por ejemplo, la latencia marcaría el tiempo transcurrido desde que el usuario mueve la cabeza, para dirigir su vista a otra parte, hasta que el sistema actualiza la imagen en pantalla. El sistema tarda más en responder a este tipo de acción que a cualquier otra, como pueda ser un disparo, debido al retardo adicional que los dispositivos de localización introducen.

2.2.3. Tridimensionalidad

El concepto de simular un mundo virtual interactivo mediante imágenes sintéticas no es nada extraño para nosotros. Todos estamos familiarizados desde hace mucho tiempo con el mundo de los videojuegos. En un videojuego, un equipo informático más o menos sofisticado es utilizado con el fin de crear y presentar en la pantalla un mundo virtual más o menos fantástico.

Pero la mayoría de los videojuegos tradicionales son puramente bidimensionales. En juegos como Sonic, Super Mario o Mortal Kombat, no existe tercera dimensión: los objetos, el usuario y la cámara se desplazan sólo en una o dos dimensiones, de izquierda a derecha y de arriba a abajo de la pantalla, pero no existe movimiento de profundidad en la escena (es decir, hacia dentro de la pantalla). Para que un sistema pueda ser llamado de realidad virtual, es necesario que proporcione además la dimensión de profundidad.

Obsérvese que es necesario que esa dimensión de profundidad exista realmente dentro de la aplicación, no bastando con que se la simule. Hay juegos, como los de la serie Hero Quest, de Sierra, que permiten el movimiento de los personajes en tres dimensiones: jugando con las perspectivas y con la decoración de las escenas, se consigue que el movimiento de una figura arriba y abajo de la pantalla pueda ser percibido por el usuario en ciertas ocasiones como un movimiento en sentido vertical, y en otras como un movimiento en sentido de profundidad. Pero en realidad no hay un tratamiento tridimensional de la escena. Los objetos móviles tienen igual tamaño independientemente de su profundidad, y la escena no está definida más que como un bitmap de fondo. Es lo que podríamos denominar dos dimensiones y media.

Un ejemplo de juego con tratamiento de tres dimensiones es Tetrix 3-D, donde el movimiento de profundidad se simula de la forma natural: alterando el tamaño del objeto a medida que se aleja. Los simuladores de vuelo, algunos juegos de guerra intergaláctica, simulaciones bélicas del estilo de Campaign o juegos como Doom o Wolf 3D, utilizan también una representación en tres dimensiones de las escenas, donde la cámara se desplaza siguiendo el punto de vista del usuario. Los objetos en este tipo de juegos están definidos bien mediante sus coordenadas tridimensionales, bien como simples mapas de bits auto-orientables, pero tienen una cierta posición en el espacio tridimensional a partir de la cual se generan las imágenes en tiempo real.

Para ver si la condición de tridimensionalidad se cumple, la pregunta podría formularse de la siguiente manera. A la hora de calcular la imagen que se presentará en la pantalla, ¿efectúa el sistema algún tipo de proyección de los objetos sobre ella? Si es así, entonces es que los objetos tienen asociada una posición "en profundidad", y aparecerán una serie de

fenómenos asociados a los mundos de tres dimensiones, como que el tamaño aparente de los objetos variará en función de su distancia del observador.

Si no es así, es decir, si el sistema no efectúa ningún tipo de proyección, entonces toda la generación de imágenes puede realizarse mediante técnicas puramente bidimensionales, y no se trata de un sistema de realidad virtual.

Es importante observar el hecho de que, una vez que se ha efectuado la proyección de la imagen sobre la pantalla, lo que pasamos a tener es una imagen en dos dimensiones, pese a lo cual somos perfectamente capaces de reconocer la naturaleza tridimensional del mundo virtual.

Un ejemplo claro es el de la TV. Las imágenes de TV son imágenes planas, de dos dimensiones; pero, a pesar de todo, podemos perfectamente, al contemplar una película, determinar cuáles objetos están situados delante de otros. Ello es debido a que una sucesión de imágenes planas (e incluso una única imagen aislada) contiene determinadas informaciones, denominadas **claves de profundidad**, que el cerebro humano emplea para deducir las profundidades relativas de los objetos.

Una de las principales claves de profundidad (de la que ya hemos hablado antes) es la variación del tamaño aparente de los objetos con la distancia al observador, efecto que se conoce con el nombre de **perspectiva lineal**. Otra es la del **paralaje**, que es el fenómeno por el cual, al desplazar lateralmente la cámara, los objetos más cercanos parecen desplazarse más que los más alejados. Al final de este mismo capítulo hablaremos más en detalle de las cuestiones relacionadas con la percepción de profundidad en el hombre.

2.2.4. Ilusión de realidad

Existen multitud de videojuegos que satisfacen completamente las tres condiciones que hasta ahora hemos expuesto. Un simulador de vuelo para PC es un sistema interactivo, capaz de sintetizar imágenes de un mundo en tres dimensiones por el que el usuario puede desplazar su aeronave a voluntad.

Pero para que un sistema pueda ser denominado de realidad virtual, debe además cumplir una cuarta condición: la de que el mundo virtual tenga apariencia de realidad. No es en absoluto necesario que el mundo virtual se parezca *al* mundo real. Basta con que *parezca real*.

Al contrario de lo que sucede con las otras tres, esta condición no permite trazar un límite estricto entre lo que es y lo que no es un sistema de realidad virtual. Existen multitud de factores que influyen en la creación de una ilusión de realidad, y los sistemas de realidad virtual emplean mecanismos diversos para lograrlo.

El que el usuario pueda sentir como real el mundo simulado va a depender tanto de factores físicos como psicológicos.

2.2.4.1. *Factores físicos*

Los factores físicos están relacionados con el "aspecto" del mundo virtual, es decir, con las percepciones (visuales, sonoras, táctiles, ...) del usuario acerca de dicho mundo. El aspecto será tanto más real cuantos más sentidos del usuario sea capaz de estimular el sistema, y cuanto más parecidas a las sensaciones reales sean esas representaciones artificiales. En particular, el sistema debe poder recrear con sus representaciones los aspectos tridimensionales del mundo virtual.

Los mecanismos utilizados por las aplicaciones de RV para poder recrear el aspecto real de los objetos son muy variados, y dependen del sentido que se pretenda estimular. Las **técnicas de estereoscopía**, los métodos de **localización de sonido** en el espacio tridimensional o la **realimentación táctil**, son todas ellas técnicas destinadas a realzar la "realidad física" de los objetos y del mundo virtual. De todas ellas hablaremos en profundidad a lo largo del libro.

2.2.4.2. *Factores psicológicos*

Los factores psicológicos están relacionados con la "naturaleza" del mundo virtual, tal como el usuario la percibe.

Entre los principales factores está el de la **interactividad**. Las posibilidades de interacción que el mundo virtual ofrezca van a influir grandemente en la aceptación psicológica por parte del usuario: la ilusión de realidad será tanto más intensa cuantas más posibilidades de interacción se ofrezcan al usuario, y cuantas más posibilidades tenga éste de influir en el estado del mundo virtual. La **facilidad de navegación**, de la que ya hemos hablado, tiene una importancia fundamental a este respecto. Para que el usuario pueda aceptar como verosímil la tridimensionalidad del mundo simulado, es un requisito indispensable el que pueda desplazarse por él.

La forma de llevar a cabo las acciones posibles también tiene su importancia. Cuanto más parecida a la del mundo real sea la *manera* de interaccionar con los objetos, mayor será la apariencia de realidad de éstos. La utilización de **periféricos de entrada** sofisticados, como los electroguantes, responde a esta ambición de simular los modos de interacción que empleamos cotidianamente.

En general, es el propio **comportamiento** del mundo virtual lo que el usuario va a valorar como verosímil o inverosímil. Esto incluye no sólo las respuestas a los estímulos del usuario, sino también el comportamiento autónomo de los objetos y, en general, todos los aspectos dinámicos del mundo virtual, es decir, cómo evoluciona. La ilusión de realidad será tanto más intensa cuanto más "se comporte" el mundo virtual como si fuera real.

Los sistemas de realidad virtual emplean otros mecanismos, como la **inmersión**, no ligados con el comportamiento del mundo virtual, pero que también influyen en la aceptación psicológica por parte del usuario. Mediante las técnicas de inmersión, se priva al usuario de la visión del mundo circundante (por ejemplo, utilizando un visiocasco); de esta manera, al perder toda posible referencia visual, la ilusión de realidad se intensifica.

Entre otros factores de carácter psicológico, podemos citar el hecho de que un **sistema** sea **multiusuario**. En este tipo de sistema, la sensación de realidad se intensifica por el hecho de que hay otros objetos en el sistema (los usuarios) con un comportamiento que parece real (porque de hecho lo es).

2.3. MECANISMOS BÁSICOS

Existen cinco mecanismos básicos habitualmente empleados en las aplicaciones de realidad virtual, que les permiten satisfacer los distintos presupuestos de nuestra definición.

2.3.1. Gráficos tridimensionales

Como ya hemos dicho, los sistemas de realidad virtual sintetizan las imágenes a partir de la definición de los objetos incluidos en el mundo virtual, y teniendo en cuenta el estado de éste y la posición del usuario.

Los sistemas de RV parten de una definición tridimensional del mundo virtual y de los objetos que lo forman, y utilizan técnicas de tratamiento de gráficos tridimensionales (3D) para generar las imágenes.

Las técnicas utilizadas en el tratamiento de gráficos 3D son relativamente maduras. Desarrolladas a partir de los primeros años 70, especialmente debido a las investigaciones llevadas a cabo en la Universidad de Utah, pronto encontraron aplicación comercial tanto en el campo de la simulación de vuelo como, más recientemente, en los programas de animación 3D para cine y TV. Como consecuencia, existen multitud de productos y componentes comerciales, tanto hardware como software, especializados en este tipo de técnica, y los principales algoritmos son bien conocidos.

Básicamente, a la hora de generar la imagen, se parte de una lista de objetos del mundo virtual, con su correspondiente definición de forma y propiedades gráficas. A partir de dicha información, se efectúa una serie de transformaciones geométricas donde se tienen en cuenta la posición del objeto y la del usuario, de forma que al proyectar el objeto en la pantalla aparezca con la adecuada perspectiva. También se efectúa una serie de cálculos para determinar el color de cada punto del objeto, tal y como lo verá el usuario, teniendo en cuenta aspectos como las fuentes de luz que hay en la escena, la forma en que el objeto refleja la luz, etc...

El resultado es una imagen de la escena vista desde la posición que el usuario ocupa, y cuya calidad será mayor o menor dependiendo de qué características gráficas de los objetos tengamos en cuenta y de los recursos de procesamiento que dediquemos a esta tarea.

En el capítulo 5 se expone en detalle el proceso de generación de las imágenes en una aplicación de RV.

2.3.2. Técnicas de estereoscopía

Hasta ahora hemos venido considerando el caso de que las imágenes calculadas de la escena se proyecten sobre una pantalla, donde el usuario las pueda contemplar. Una vez efectuada la proyección, lo que el usuario percibe es una imagen bidimensional. Sin embargo, las transformaciones geométricas y el resto de los cálculos realizados durante el proceso de generación de los gráficos 3D aseguran que la sucesión de imágenes generada va a contener una serie de claves de profundidad, mediante las que el usuario va a poder percibir la naturaleza tridimensional de la escena.

El usuario puede continuar percibiendo la información de profundidad porque dichas imágenes bidimensionales se han calculado proyectando la información de tres dimensiones que componía la escena, con lo que contienen la serie de claves de profundidad. Pero la imagen en sí es plana.

Una forma de acentuar la "realidad" de la imagen consiste en utilizar técnicas de estereoscopía. Las técnicas de estereoscopía permiten al usuario no sólo percibir las claves de profundidad, sino además ver las imágenes efectivamente en relieve.

El ser humano es capaz de ver en tres dimensiones porque las imágenes que percibe con cada ojo son ligeramente distintas. Aun cuando vea la misma escena con ambos ojos, la perspectiva de cada ojo es ligeramente diferente, debido a la diferente posición relativa de cada ojo con respecto a los objetos. Esto permite al cerebro comparar las dos imágenes recibidas y deducir, a partir de las diferencias relativas, la profundidad a que se encuentra cada parte de un objeto.

Es importante entender que la visión en tres dimensiones es posible porque el ser humano, al tener los ojos situados en la parte frontal del cráneo, percibe dos perspectivas distintas de una *misma* escena. Otras especies animales, cuyos ojos están situados lateralmente y apuntan en direcciones muy diferentes, no pueden percibir en tres dimensiones, porque las imágenes que perciben a través de los ojos son completamente distintas y no se puede extraer por comparación de las mismas la información de profundidad.

Las técnicas de estereoscopía utilizan este mecanismo para hacer que el sistema proporcione una visión en relieve. Para ello, es necesario que el sistema de realidad virtual calcule por separado cuál es la imagen de la escena que el usuario percibiría por cada ojo, y que luego presente ambas imágenes de manera simultánea, cada una por un ojo. En el capítulo 3 explicaremos cuáles son las técnicas de estereoscopía empleadas por los sistemas de RV.

2.3.3. Simulación de comportamiento

En la mayoría de los sistemas de RV (excepción hecha de las aplicaciones de paseo virtual), el estado del mundo virtual cambia con el tiempo, de acuerdo con las acciones del usuario, con otros estímulos externos, y con el propio comportamiento que para los objetos del mundo virtual se haya definido (figura 2.4).

Al igual que sucedía con la representación del mundo virtual, esta simulación se realiza de manera sintética: no se tiene precalculada la evolución del estado del mundo virtual, sino que se va calculando en tiempo real, de acuerdo con los factores mencionados.

La tarea de simulación en un sistema de RV puede tener dos cometidos diferentes. Por un lado, la simulación puede emplearse para remedar el comportamiento de algún objeto o sistema del mundo real. Es el caso, por ejemplo, de los simuladores de vuelo, donde lo que se busca es hacer que el sistema se comporte como su equivalente real, para entrenar al piloto en condiciones lo más parecidas posible a las que luego va a encontrar.

En estos casos, la apariencia de realidad del sistema viene inmediatamente determinada por la fidelidad con la que el sistema de RV reproduzca el comportamiento del sistema real. Puesto que el primero es un remedo del segundo, el usuario siempre tiene un marco de referencia al que acudir para valorar la "realidad" del sistema de RV.

En aquellos otros casos donde los objetos simulados no tengan un equivalente demasiado directo en el mundo real, la apariencia de realidad del sistema vendrá dada por factores más subjetivos. En general, un comportamiento rico de los objetos, la coherencia del mismo o un alto grado de interactividad contribuirán a aumentar el realismo del sistema.

Resulta curioso observar que esa ilusión de realidad se intensifica cuanto más fantástico es el mundo virtual: cuantas menos equivalencias con el mundo real, menos referencias tiene

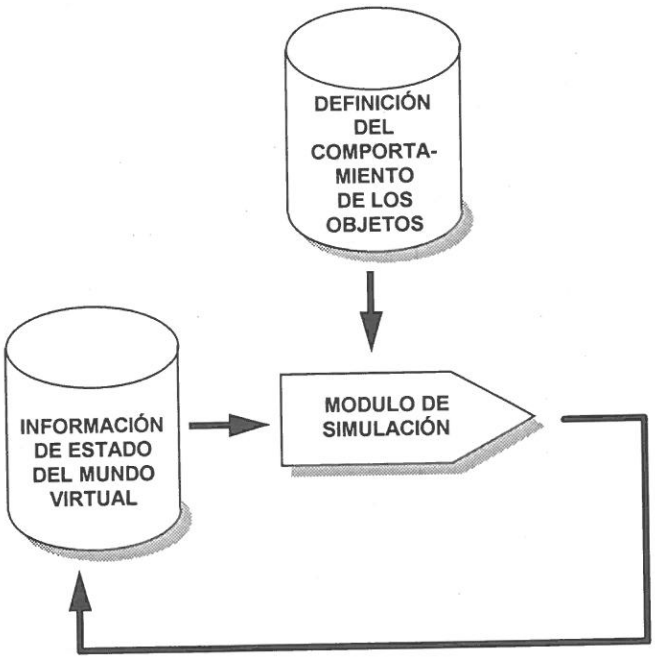

Figura 2.4. Simulación de comportamiento

el usuario para cuestionar la validez de lo que percibe, y pierde importancia la calidad de la imagen en sí. Aunque este efecto tiene dos límites. El primero es que las reglas que gobiernan el mundo virtual deben ser entendibles para el usuario. Si son tan complejas que el usuario no puede llegar a percibir la mecánica que subyace, la ilusión de realidad se destruye porque no comprende lo que ve.

El segundo límite es que ciertos mecanismos físicos de carácter básico deben conservarse, especialmente en lo que se refiere a la interacción con el sistema. Estudios experimentales han mostrado, por ejemplo, que el usuario es particularmente sensible a las rotaciones de la información de entrada y de salida: difícilmente se puede percibir como real, por ejemplo, un sistema donde al volver la cabeza hacia la izquierda, la imagen se desplace hacia abajo. Al igual que no resulta posible controlar un sistema donde, para desplazarse hacia adelante, haya que empujar el joystick hacia la derecha. Otro tipo de cambios, como las traslaciones o los cambios de escala, suelen ser asumidos por el usuario sin mayor problema.

En cualquier caso, el comportamiento del mundo virtual se traduce, al final, en cambios en la representación (gráfica o de otro tipo) de los objetos. Los objetos podrán variar su posición, cambiar de forma o emitir sonidos. Pero es necesario, obviamente, que algún tipo de cambio se produzca. Un comportamiento de un objeto que no tenga traducción en algo que el usuario pueda percibir es algo completamente superfluo.

Dicha traducción no necesita ser inmediata. Es posible que un cambio en el estado de un objeto no se perciba hasta pasado un cierto tiempo (como sucede, por ejemplo, con el nivel de energía de un monstruo, cuyas disminuciones no se perciben hasta que el nivel de energía del monstruo se agota y éste muere). Pero, de nuevo, la regla de traducción tiene que ser comprensible.

Dependiendo del tipo de aplicación, la simulación de comportamiento puede ser más o menos compleja, y tener mayor o menor importancia. Así, en una aplicación de diseño arquitectónico, los requerimientos de simulación pueden ser pequeños, mientras que es necesario un tratamiento gráfico de alta calidad. En las aplicaciones de visualización científica, por el contrario, es posible que la mayor parte de la capacidad de proceso del sistema se dedique a la simulación de magnitudes físicas, teniendo la calidad de la imagen una importancia relativamente menor.

2.3.4. Facilidades de navegación

Las facilidades de navegación son los mecanismos puestos a disposición del usuario para poder variar su posición u orientación dentro de una escena.

Existen dos tipos de dispositivos que pueden utilizarse para implementar las facilidades de navegación: **dispositivos de control** y **dispositivos de localización**. Nuestro sistema hipotético del capítulo 1 incluía un ejemplo de cada uno de estos dos tipos.

Las **dispositivos de control** permiten al usuario indicar de manera expresa sus deseos en lo que a la navegación se refiere. Haciendo uso de un joystick o de las teclas de cursor de un ordenador, puede señalar al sistema la dirección en la que desea desplazarse. También

podría utilizar las teclas de control para variar su orientación dentro de una escena, sin moverse de posición.

Los **dispositivos de localización**, por el contrario, permiten al sistema determinar los deseos del usuario de una manera implícita. Utilizando un dispositivo que permita conocer la posición y orientación reales del usuario, el sistema puede variar de acuerdo con ellas la posición y orientación dentro del mundo virtual. Así, cuando el usuario vuelva la cabeza hacia la izquierda, el sistema detectará el hecho y desplazará hacia la derecha la imagen de la pantalla. Si el usuario da unos pasos hacia adelante, el sistema actualizará correspondientemente su posición dentro del mundo virtual.

Como vemos, los dispositivos de localización permiten un control mucho más natural de las facilidades de navegación. Presentan, sin embargo, algunas desventajas (de las que hablaremos en detalle en el capítulo 4), por lo que es raro encontrar aplicaciones de RV donde la navegación se controle haciendo uso exclusivamente de dispositivos de localización. Normalmente, la navegación se regula bien mediante dispositivos de control solamente, bien mediante una combinación de ambos tipos de dispositivo. Una solución muy utilizada consiste en emplear algún dispositivo de control para regular los desplazamientos, y un dispositivo de localización para las orientaciones, tal como se hacía en nuestro ejemplo del capítulo 1.

Hay sistemas, como los inmersivos, donde el uso de los dispositivos de localización es obligatorio. Puesto que los sistemas inmersivos tienen por objeto inducir en el usuario la sensación de que se haya rodeado por el mundo virtual, es necesario que la imagen se actualice cada vez que el usuario gire la cabeza, lo cual excluye el control expreso de la orientación.

Independientemente del tipo de dispositivo que se emplee, el incluir en un sistema de RV facilidades de navegación tiene dos consecuencias. La primera es que, dependiendo de la posición que el usuario ocupe, la información gráfica, sonora, etc... del mundo virtual que el sistema le presente será una u otra. Así, por ejemplo, el sistema presentará las imágenes al usuario desde el adecuado punto de vista. Los objetos situados "delante del usuario" aparecerán en la pantalla, mientras que los situados "detrás" no lo harán. De la misma manera, los sonidos de los sucesos más próximos al usuario tendrán mayor volumen que los de los sucesos distantes, los objetos aparecerán en la pantalla tanto más pequeños cuanto más alejados se encuentren, etc...

La segunda consecuencia es que la propia posición del usuario puede afectar al estado del sistema, si es que éste realiza alguna simulación de comportamiento. Es posible, por ejemplo, que determinados objetos reaccionen ante la proximidad del usuario, o que determinadas variables globales de estado se vean afectadas.

Vemos pues que, para un sistema de realidad virtual, la posición del usuario tiene un doble significado: por un lado, se trata de una variable de estado más, que hay que tener en cuenta a la hora de efectuar la simulación del mundo virtual; por otro lado, la posición del usuario determina cómo se deben representar las escenas (figura 2.5).

La posibilidad de navegación puede estar (y normalmente lo estará) sujeta a una serie de restricciones, impuestas por el desarrollador de la aplicación. Algunas de ellas pueden ser inherentes a la propia lógica del mundo virtual, como el hecho de que no pueda atravesarse una pared (por muy virtual que sea) o que los movimientos estén restringidos al plano de

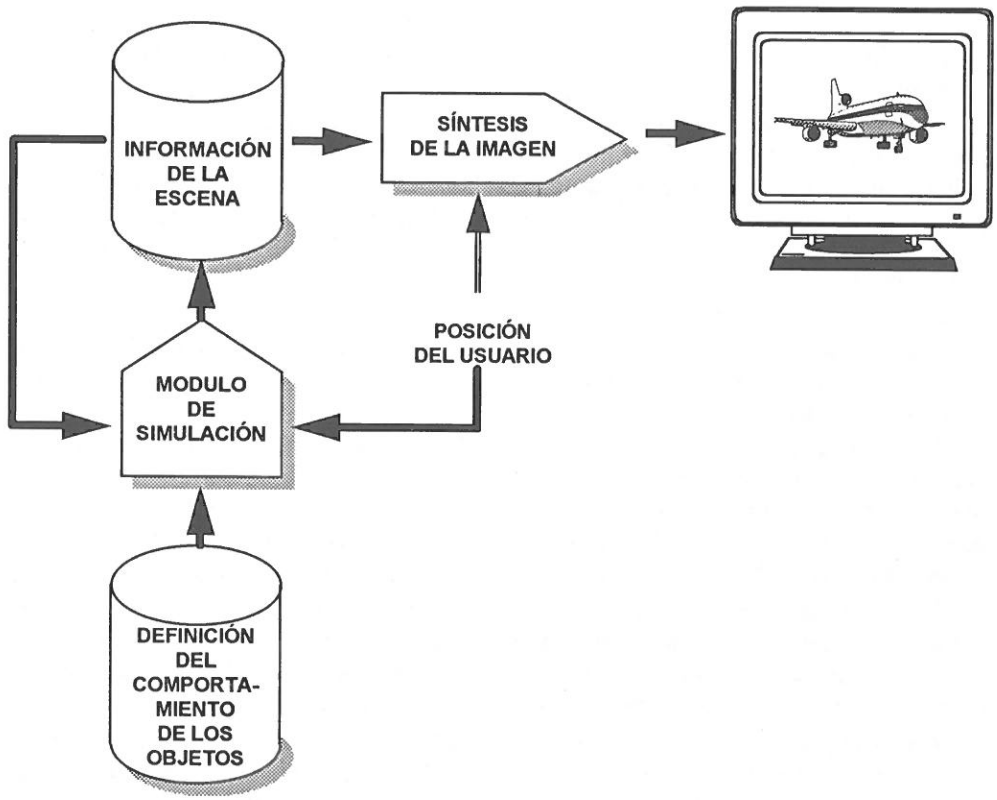

Figura 2.5. Influencia de la posición del usuario

tierra en el caso de un simulador de conducción. Pero en otras ocasiones pueden imponerse restricciones arbitrarias, por ejemplo para optimizar el tratamiento gráfico de las escenas. Sería el caso, por ejemplo, de un simulador de conducción donde el vehículo tenga prohibido salir fuera de la carretera.

2.3.5. Técnicas de inmersión

Las técnicas de inmersión consisten en aislar al usuario de los estímulos procedentes del mundo real circundante, con el fin de acentuar la apariencia de realidad de la aplicación. Este efecto se produce porque, al quedar privado de las sensaciones procedentes del mundo real, se pierde la referencia con la que poder comparar las sensaciones que el mundo virtual produce.

Los sistemas de RV se pueden clasificar en tres categorías diferentes, según el grado de inmersión que proporcionan: sistemas inmersivos, donde el usuario utiliza un dispositivo de

visualización que le aísla completamente del entorno, como puede ser un visiocasco; sistemas proyectivos, donde la inmersión se consigue encerrando al usuario en un recinto en el que se emplean una o más pantallas como si fueran ventanas hacia un mundo virtual situado en el exterior del recinto; y sistemas no inmersivos, o de sobremesa, donde el mundo virtual se muestra en una pantalla normal de ordenador, sin impedir al usuario la visión del mundo real circundante.

En los sistemas inmersivos puros, se busca provocar en el usuario la sensación de estar situado dentro del mundo virtual (de estar inmerso en él). En consecuencia, el sistema necesita incorporar algún tipo de dispositivo de localización, con el fin de detectar hacia dónde el usuario dirige su vista. Generalmente, los dispositivos de localización contribuyen grandemente a aumentar la latencia del sistema, lo que puede provocar mareos en los usuarios después de un cierto tiempo de uso continuado (ver más abajo).

En los sistemas proyectivos, lo que se busca es inducir en el usuario la idea de que está recorriendo el mundo virtual dentro de algún tipo de vehículo. En consecuencia, es perfectamente coherente que el usuario perciba los objetos situados en el interior del vehículo, puesto que todo el vehículo se desplaza simultáneamente por el mundo virtual.

2.4. CARACTERIZACIÓN DE LOS SISTEMAS DE RV

Los cinco mecanismos expuestos en la sección precedente nos van a permitir caracterizar los sistemas de realidad virtual, y trazar los límites entre lo que es y lo que no es una aplicación de RV. Aunque, como veremos, dichos límites son un tanto difusos.

Un sistema que haga uso de los cinco mecanismos mencionados será, sin lugar a dudas, un sistema de RV. Pero eso no quiere decir que si uno de los cinco mecanismos está ausente, el sistema no sea de RV.

Existe una cierta gradación en cuanto a la importancia de los distintos mecanismos. Podríamos decir que las facilidades de navegación y la generación de gráficos 3D son requisitos imprescindibles para poder considerar que un cierto sistema es de RV.

En cuanto a los otros tres mecanismos, existen aplicaciones donde puede estar ausente la simulación de comportamiento (como sucede en los paseos virtuales), o que no sean inmersivas (como es el caso de los sistemas de RV de sobremesa) o donde no se empleen técnicas de estereoscopía (como en algunos sistemas proyectivos).

En consecuencia, es posible que uno de dichos mecanismos esté ausente de un sistema de RV. Ahora bien, sería necesario pensárselo dos veces antes de calificar de RV a una aplicación en la que dos de estos tres mecanismos faltaran de manera simultánea.

A la vista de lo anterior, podemos juzgar (aproximadamente) si un sistema se puede calificar como de realidad virtual. No es posible establecer reglas exactas, aunque sí podemos excluir algunos tipos de sistemas: la mayoría de los videojuegos no pueden considerarse sistemas de RV, porque simulan mundos virtuales que no tienen tres dimensiones; los programas de animación, que se utilizan para construir escenas que sí son tridimensionales, tampoco pueden considerarse aplicaciones de RV, porque carecen de simulación en tiempo real del comportamiento y de capacidad inmersiva.

Los sistemas de sobremesa que no emplean técnicas de estereoscopía no son tampoco considerados, en general, como sistemas de RV. Este es el caso, por ejemplo, de los videojuegos que emplean gráficos 3D, como el Virtua Fighter de Sega o el simulador de conducción Ridge Racer de Namco. Sin embargo, obsérvese lo delgada que es la frontera que separa a los sistemas de RV de los videojuegos: añadir capacidad estereoscópica a uno de estos juegos no es en modo alguno difícil, y eso sólo bastaría para poder considerarlos auténticos sistemas de RV. Si el realismo de la aplicación se acrecentara utilizando otras técnicas distintas de la estereoscopía, por ejemplo técnicas de simulación de la inercia mediante plataformas móviles para el caso de un simulador de conducción, empezaría a ser difícil no considerar al sistema como de RV.

2.5. ARQUITECTURA HARDWARE

2.5.1. Diagrama general

Un sistema de realidad virtual estará, básicamente, constituido por:

- Una serie de dispositivos de entrada, mediante los que el usuario comunica sus órdenes al sistema, o que el sistema utiliza para obtener información del entorno, por ejemplo la posición del usuario.
- Una serie de dispositivos de salida, que el sistema emplea para proporcionar al usuario información acerca del mundo virtual y provocar en él determinadas sensaciones.
- Una estación de procesamiento, encargada de realizar las tareas de simulación, y el control de la entrada y salida de los datos.

2.5.2. Dispositivos de entrada

El usuario se comunica con el sistema a través de dos tipos de dispositivos, como ya antes hemos comentado:

- Dispositivos de localización.
- Dispositivos de control.

Los dispositivos de localización sirven para medir la posición y orientación de cualquier objeto o parte del mismo en el espacio de tres dimensiones. Su principal aplicación en los sistemas de RV es la de medir la dirección en la que el usuario está mirando, aunque también se suelen emplear en conexión con los electroguantes, para determinar la posición de la mano. Los dispositivos de localización se clasifican, según la tecnología utilizada, en dispositivos electromagnéticos, dispositivos mecánicos y dispositivos ultrasónicos.

Los dispositivos de control, por su parte, permiten al usuario comunicar sus órdenes al sistema de una manera explícita. Existen tres tipos básicos de órdenes: comandos de navegación, comandos de interacción con los objetos y comandos de manipulación del estado del sistema. Los sistemas de RV emplean tanto dispositivos convencionales (ratón, teclado,

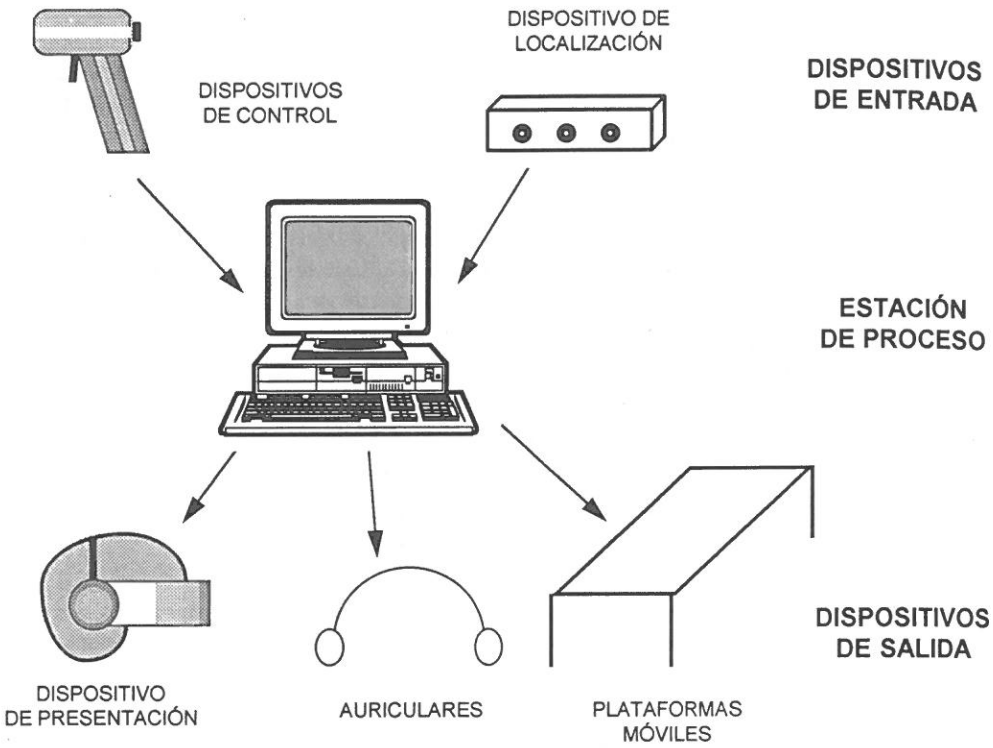

Figura 2.6. Configuración de un sistema de RV

joystick) como periféricos más o menos sofisticados, como pueden ser los ratones y joysticks 3D, electroguantes o dispositivos bioeléctricos.

De todos estos dispositivos hablaremos en el capítulo 4.

2.5.3. Dispositivos de salida

El sistema proporciona información al usuario mediante diversos dispositivos de salida, con los que puede representar el estado del mundo virtual. Dependiendo del sentido que se busque estimular, los dispositivos hoy en día empleados en aplicaciones de RV se clasifican en:

- Dispositivos de presentación.
- Dispositivos de sonido.
- Dispositivos de realimentación táctil y cinestésica.
- Dispositivos móviles.

Los dispositivos de presentación tienen como cometido mostrar al usuario la información de carácter gráfico generada por el sistema. Los sistemas proyectivos y de sobremesa suelen hacer uso de dispositivos más o menos convencionales, como monitores y proyectores, pudiendo emplear dispositivos especiales, como las gafas estereoscópicas, para permitir la visión en relieve de las escenas.

Las aplicaciones inmersivas, por su parte, suelen hacer uso de periféricos especializados, como los visiocascos o los sistemas binoculares, que impiden la visión del mundo circundante al mismo tiempo que proporcionan visión estereoscópica. Este tipo de dispositivo incorpora siempre, además, el correspondiente dispositivo de localización.

El sonido tiene una gran importancia a la hora de dotar de apariencia de realidad a un sistema de RV. El tratamiento del sonido en muchas aplicaciones de RV suele ser, sin embargo, bastante convencional. Pero existen dispositivos especializados en la generación de sonido 3D que permiten a los usuarios localizar la procedencia de las fuentes de sonido.

Los dispositivos de realimentación táctil y cinestésica permiten estimular el sentido del tacto y las resistencias mecánicas de los objetos. Este tipo de dispositivo encuentra su mayor utilidad en las aplicaciones de tele-robótica y de simulación científica.

Los dispositivos móviles se emplean únicamente en sistemas de carácter proyectivo o inmersivo, y tienen por objeto someter al usuario a movimientos lo más reales posible dentro del mundo virtual. Utilizando una plataforma móvil, por ejemplo, se pueden simular las traslaciones, los giros e incluso las aceleraciones del usuario en su navegación por el mundo virtual.

Hablaremos de todos estos dispositivos en el capítulo 3. El tema de la síntesis de imágenes se explica en el capítulo 5.

2.5.4. Estación de proceso

La estación de proceso será la encargada de gestionar la base de datos que describe el estado del mundo virtual, interpretar las órdenes del usuario, detectar la posición de éste, calcular la evolución del mundo virtual de acuerdo al comportamiento de los objetos que lo componen, y presentar al usuario la información gráfica o de cualquier otro tipo que sea necesaria.

Para integrar un sistema de RV, cualquier ordenador de propósito general es perfectamente válido, sólo con añadirle el hardware y software especializado que las aplicaciones de RV requieren.

Será necesario añadir, por ejemplo, tarjetas gráficas especializadas en la generación de imágenes 3D, tarjetas de localización de sonido tridimensional o controladores de dispositivos de realimentación táctil. También determinados dispositivos de entrada, como los de localización o los electroguantes, requerirán la inclusión de controladores hardware específicos.

Con respecto al software, será imprescindible utilizar la capa de bajo nivel formada por los drivers de los distintos dispositivos instalados. A nivel más alto, toda la lógica de simulación puede escribirse directamente en cualquier lenguaje convencional u orientado a

objetos, como C, Pascal o C++. Lo normal, sin embargo, es emplear alguno de los diversos sistemas comerciales de desarrollo de aplicaciones de realidad virtual, los cuales proporcionan su correspondiente entorno de ejecución. De esa forma, la tarea de programación de las aplicaciones se simplifica sobremanera, puesto que las librerías que esos sistemas de desarrollo proporcionan incluyen funciones para gestionar movimientos de objetos, lanzamiento de animaciones, creación y destrucción de objetos, cambios de escala, rotaciones, etc...

Eso no significa que la programación de las aplicaciones sea trivial. Por el contrario, las librerías de los sistemas de desarrollo de aplicaciones de RV contemplan únicamente los tipos más básicos de comportamiento, por lo que aun utilizándolas será necesaria una labor de programación considerable para dotar a los objetos de la funcionalidad requerida. En particular, los aspectos más matemáticos de la simulación, especialmente en aplicaciones de visualización científica, deberán ser programados a medida.

El tema de la estación de proceso de un sistema de RV y del desarrollo de aplicaciones se tratan, respectivamente, en los capítulos 6 y 7 del presente libro.

2.6. ALGUNOS PARÁMETROS DE IMPORTANCIA

2.6.1. Velocidad de generación de las imágenes

Una de las características esenciales de cualquier sistema de RV es el número de imágenes por segundo que el sistema es capaz de calcular. Dependerá fundamentalmente de los requerimientos gráficos que la aplicación imponga: el número de objetos que compongan la escena, el nivel de detalle con el que dichos objetos se describan, y la calidad global de imagen que se especifique (tipo de sombreado, tipo y número de las fuentes de luz, etc...).

La velocidad de generación de las imágenes puede ser síncrona o asíncrona. En los **sistemas síncronos**, se fija de antemano una tasa de generación de imágenes, y la carga del sistema se calcula para que dicha tasa se alcance. Así, si se fija una velocidad de 10 imágenes por segundo, el sistema dispondrá de 100 mseg para hacer todos los cálculos requeridos. Una vez que la nueva imagen ha sido calculada, el sistema espera a que transcurran completamente los 100 mseg antes de presentarla en la pantalla. En caso de que la carga de trabajo creciera y el sistema no tuviera tiempo de completar los cálculos requeridos, se recurre a determinadas **técnicas de gestión del nivel de detalle**, que permiten modificar dinámicamente el nivel de detalle con el que los objetos se representan.

En los **sistemas asíncronos**, el sistema calcula la nueva imagen y, tan pronto como ha terminado, la presenta en pantalla y comienza con la siguiente. El esquema tiene la ventaja de que no hay tiempos muertos, ni tampoco temporizaciones estrictas con las que el sistema tenga que cumplir. Sin embargo, con este sistema, el tiempo transcurrido entre una imagen y la siguiente varía de forma continua, lo que tiene varias desventajas: aparecen efectos ópticos indeseados, no existe forma de sincronizar las animaciones con el movimiento y, en los sistemas de entrenamiento, se pueden originar reflejos condicionados falsos. El ejemplo típico es el del piloto que, en el simulador de vuelo, sabe que la base enemiga está próxima,

aunque no la vea, simplemente porque el sistema se ralentiza (debido a la gran cantidad de objetos que tiene que procesar).

La velocidad mínima admisible de generación de imágenes es de 10 imágenes por segundo. Por debajo de dicha tasa, no sólo no existe ilusión de realidad que valga, sino que el sistema se vuelve realmente insufrible. Entre 10 y 20 imágenes por segundo, la calidad es aceptable, y puede conseguirse una buena apariencia de realidad cuidando el resto de los aspectos del sistema. Por encima de 20, la calidad es buena, aunque algunos efectos ópticos (de importancia, por otra parte, relativa) no desaparecen hasta que se alcanzan las 30 imágenes por segundo.

El problema de la velocidad de generación de imágenes se debe a que existe un compromiso entre ésta y la calidad de la imagen. Para un sistema con una potencia de generación de gráficos 3D determinada, el aumentar al doble la tasa de generación de imágenes implica reducir a la mitad el nivel de detalle de los objetos. Como, generalmente, se trata de conseguir la mayor calidad de imagen posible, la mayor parte de las aplicaciones mantienen la tasa de generación de imágenes tan baja como pueden.

2.6.2. Latencia

Otra de las características básicas de un sistema de RV es su **latencia**. Como ya hemos indicado, al hablar de la interacción, la latencia es el tiempo transcurrido entre el instante en que el usuario efectúa una determinada acción y el momento en que el sistema actualiza la información que entrega al usuario. Generalmente, la latencia se expresa con respecto a la información de carácter gráfico, puesto que es ésta la información que más tiempo tarda en ser generada por el sistema.

La importancia de este parámetro viene dada porque los valores altos de latencia no sólo destruyen la apariencia de realidad de la aplicación y dificultan el manejo de ésta por parte del usuario, sino que incluso pueden llegar a ocasionarle mareos, fenómeno éste conocido con el nombre de "mareo de simulación". Por dar una orientación cuantitativa, la latencia máxima permitida en los simuladores de vuelo comerciales es de 150 mseg.

Si la latencia es muy alta, pueden aparecer oscilaciones, debidas a sobrecompensaciones realizadas por el usuario. En el caso de un simulador de vuelo, si el usuario detecta que el avión deriva a la derecha, tratará de corregir hacia la izquierda el rumbo. Si el sistema tarda demasiado en responder, el usuario tenderá a realizar una sobrecompensación, con lo que el avión se desviará en exceso hacia la izquierda. El usuario responde entonces compensando de nuevo hacia la derecha y, debido al retardo del sistema, tenderá de nuevo a sobrecompensar, produciéndose así la oscilación.

Este límite de oscilación (es decir, la latencia máxima admisible desde el punto de vista del control por el usuario) varía mucho de una a otra aplicación. En los simuladores de vuelo, este límite está en torno a los 250-300 mseg, mientras que hay otras aplicaciones donde, debido a la naturaleza de la interacción con el usuario, el fenómeno de la oscilación no puede llegar a producirse.

Por debajo del límite de oscilación, el usuario puede controlar el sistema sin mayores problemas, pero para valores de latencia superiores a 100 mseg, la calidad subjetiva del sistema (según la percibe el usuario) se degrada rápidamente debido al excesivo tiempo de respuesta. En estas condiciones, la ilusión de realidad se pierde completamente.

Por debajo de los 100 mseg, la calidad del sistema es tolerable (desde el punto de vista del tiempo de respuesta), pero con valores de latencia superiores a unas pocas decenas de milisegundos puede aparecer el fenómeno del "mareo de simulación".

El mareo de simulación es bastante conocido entre los pilotos (el 30% de los que utilizan simuladores de vuelo han sufrido alguna vez la experiencia). La causa de este fenómeno está en que el hombre percibe los movimientos del cuerpo mediante las terminaciones nerviosas de los canales semicirculares situados en el oído interno. Cuando los movimientos percibidos por el oído interno no se corresponden con la información que los ojos transmiten, aparece el mareo.

Es lo que sucede, por ejemplo, cuando uno lee dentro de un vehículo en movimiento: la vista está fija, mientras que el oído interno indica que nos estamos moviendo (¿cuántas veces le ha dicho usted a su hijo que mire por la ventanilla del coche para no marearse?). No se sabe a ciencia cierta la razón de por qué el cuerpo humano reacciona así, aunque es bastante posible que se trate de un acto de carácter defensivo: una de las posibles causas de que la vista indique algo distinto de lo que aprecia el sentido del equilibrio es la intoxicación alimentaria; en consecuencia, el cuerpo reacciona expulsando fuera los alimentos ingeridos.

Este fenómeno aparece siempre que hay disparidad entre la información detectada por los sentidos de la vista y el equilibrio. Existen dos circunstancias típicas en los sistemas de RV bajo las que esta disparidad puede producirse.

La primera circunstancia es cuando la imagen indica un movimiento, mientras que nuestro cuerpo está quieto. Esta situación es típica en los sistemas de sobremesa y proyectivos, donde los movimientos de las imágenes que vemos en pantalla no tienen ninguna correspondencia con movimientos del cuerpo.

En los sistemas de sobremesa es difícil que aparezca el mareo, puesto que el usuario tiene siempre a la vista referencias estáticas. Sin embargo, cualquiera que haya jugado a un juego como Doom en un 486 medianamente potente podrá certificar que también en un sistema de sobremesa puede aparecer el mareo cuando la atención se fija demasiado en la pantalla y la velocidad de movimiento de las imágenes es alta.

En los sistemas proyectivos, se pierde la referencia estática, porque inconscientemente tendemos a esperar que la cabina completa se mueva al compás de las imágenes, lo que facilita la aparición del mareo. Puede contrarrestarse este efecto colocando la cabina sobre una plataforma móvil que se incline a un lado u otro de acuerdo con lo que la imagen muestre. Sin embargo, esta compensación parcial casi nunca es perfecta.

En cualquier caso, en los sistemas de sobremesa y proyectivos, el usuario puede evitar la aparición del mareo apartando su vista de la pantalla y fijándola sobre un objeto estático.

La segunda circunstancia en que el mareo de simulación puede producirse es típica de los sistemas inmersivos, y se produce cuando efectuamos un cierto movimiento y la imagen

tarda en responder. Como ya hemos indicado, el fenómeno se agrava cuando el sistema inmersivo utiliza dispositivos de localización con un alto grado de latencia.

En este caso de los sistemas inmersivos, no existe posibilidad de compensación, ni tampoco tiene el usuario la oportunidad de apartar su vista de la imagen, porque la imagen del mundo virtual es lo único que puede ver. En consecuencia, la única solución estriba en mantener la latencia lo más baja posible.

Este fenómeno de la latencia es uno de los que más quebraderos de cabeza causan a la hora de realizar el diseño de un sistema de RV. Cada uno de los elementos hardware y software empleados interviene en el cálculo de la latencia total del sistema, y es necesario verificar con cuidado ésta a la hora de tomar decisiones en cuanto a qué arquitectura o qué elementos utilizar.

2.6.3. Compromisos

El diseño de un sistema de realidad virtual es, fundamentalmente, una cuestión de compromisos. Esos compromisos afectan, principalmente, a tres factores:

- Velocidad de síntesis.
- Calidad de la imagen.
- Latencia.

La estación de proceso que elijamos, junto con todos los módulos accesorios de procesamiento gráfico, determinará la capacidad que nuestro sistema tenga para generar imágenes, es decir, la potencia gráfica.

Para una potencia gráfica determinada, el diseñador del sistema debe elegir entre aumentar la calidad de las escenas o aumentar la velocidad de síntesis, es decir, el número de escenas generadas por segundo.

Cuanto mayor sea la calidad de las escenas, es decir, cuantos más objetos haya en la misma y más detallada sea la representación gráfica de éstos, más potencia gráfica será necesaria para generar una única escena, con lo que el número de escenas por segundo disminuye. Cuantas más escenas por segundo queramos generar, menor será la calidad de éstas.

Si queremos aumentar la calidad de las escenas sin que la velocidad de síntesis baje de un cierto límite, entonces no tenemos otro remedio que aumentar la potencia gráfica del sistema. Pero eso significa normalmente (debido a las limitaciones presupuestarias) que el presupuesto dedicado a los aspectos gráficos del sistema aumenta en detrimento de otras partes del mismo, como pueden ser las tareas de simulación o de localización. En este caso, es posible que la latencia del sistema aumente hasta superar los límites previamente fijados.

Dada la inexistencia de estándares, la proliferación de diferentes productos comerciales y la relativa novedad de esta tecnología, esta labor de diseño del sistema suele representar la fase más delicada, por contraposición a otro tipo de campos donde las técnicas básicas son más o menos estables.

En el capítulo 6 veremos, al hablar de las estaciones de proceso, algunos de los procedimientos y arquitecturas más comúnmente utilizados en aplicaciones de RV para incrementar la potencia de proceso de los sistemas. En el capítulo 7 hablaremos también más en detalle de los principales criterios utilizados a la hora de diseñar un sistema de RV.

2.7. CLAVES DE PROFUNDIDAD

Como ya hemos comentado anteriormente, las imágenes generadas por un sistema de RV contienen una serie de claves de profundidad, debido al hecho de que han sido obtenidas por proyección de un modelo tridimensional abstracto de la escena. Dichas claves de profundidad permiten al cerebro deducir la posición "en profundidad" que los distintos objetos tienen dentro de aquélla.

En este proceso de percepción de la profundidad intervienen mecanismos tanto físicos como psicológicos.

2.7.1. Claves físicas

Las claves físicas de profundidad son cuatro diferentes: disparidad binocular, paralaje, convergencia y acomodación.

2.7.1.1. Disparidad binocular o estereoscopía

Cuando el hombre contempla una determinada escena, las imágenes que se perciben de los objetos son diferentes para cada ojo, como ya se ha comentado más arriba. Ello se debe a la separación existente entre los dos ojos, que hace que los rayos de luz lleguen con un ángulo diferente para un ojo que para el otro. Esta diferencia entre las imágenes es usada por el cerebro para deducir la información de profundidad sobre los objetos.

El efecto de percepción de la profundidad por este medio decrece muy rápidamente con la distancia. Para los objetos próximos, las diferencias de ángulo entre los rayos de luz que van a cada ojo son relativamente importantes, porque la distancia no es muy grande con respecto a la separación de los ojos. Sin embargo, a partir de los 10 metros (aproximadamente), la distancia interocular empieza a ser despreciable con respecto a la distancia del objeto, y los rayos de luz que van a los dos ojos son casi paralelos. En consecuencia, el cerebro empieza a no poder detectar ninguna diferencia entre las dos imágenes del objeto y pierde, por consiguiente, la capacidad de determinar su profundidad a partir de este mecanismo.

2.7.1.2. Paralaje

Es el movimiento relativo que percibimos en los objetos al desplazar la cabeza a un lado y a otro, o al movernos a uno y otro lado de la escena. Cuando nos desplazamos o cambiamos nuestra orientación, los objetos situados más cerca parecen desplazarse más que los situados más lejos, lo que permite al cerebro inferir las distancias relativas de unos y otros. El efecto

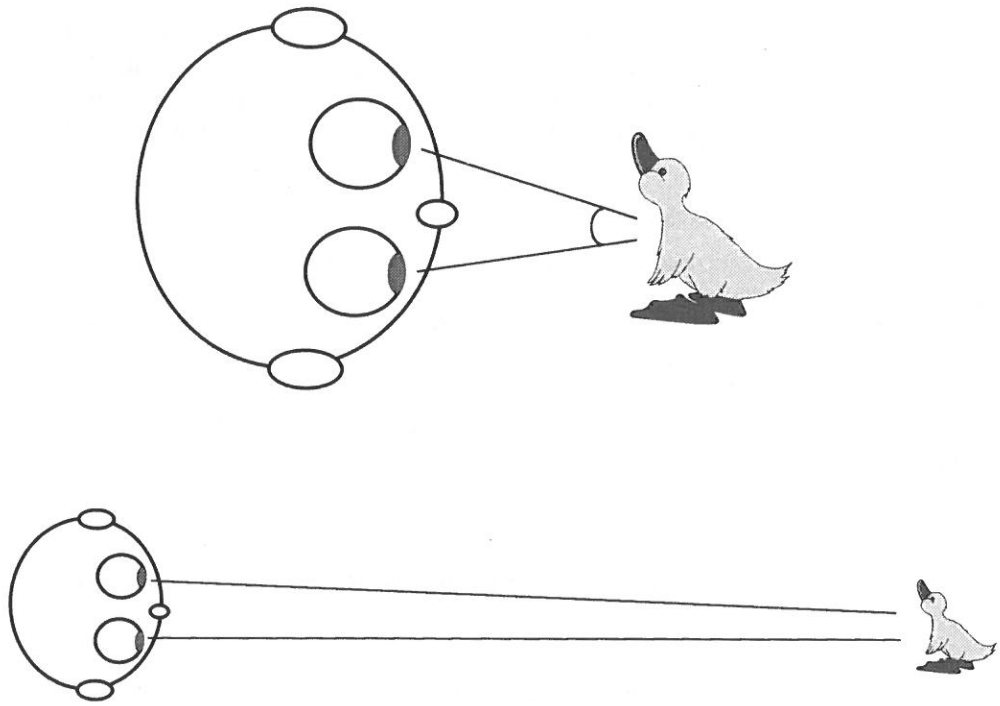

Figura 2.7. Disminución de la disparidad binocular con la distancia

es tanto más acusado cuanto más parecida es la dirección en la que los objetos se encuentran, siendo máximo cuando los objetos se superponen.

2.7.1.3. Convergencia

El proceso de la visión estereoscópica se basa en el hecho de que el cerebro "funde" en una sola las imágenes que llegan a los dos ojos. Pero esta capacidad de fundir las imágenes no se produce simultáneamente para todas las distancias. Para comprobar este hecho basta con levantar el dedo índice de una mano a unos 20 o 30 cm de los ojos, y el dedo índice de la otra mano a otros 20 o 30 cm más, detrás del primero. Si intenta fijar su vista sobre el dedo más cercano, verá doble el dedo más alejado, y viceversa.

Cuando queremos fijar nuestra atención sobre un objeto, lo que hacemos es modificar el ángulo que forman nuestros dos ojos, para que así se funda la imagen correspondiente a *ese* objeto.

Esa variación que se produce en el ángulo que forman los ejes de visión de los dos ojos cuando dirigimos nuestra atención hacia objetos situados a diferentes profundidades es lo

que se conoce con el nombre de convergencia. Para objetos próximos, los ojos se inclinan hacia dentro, mientras que para objetos lejanos los dos ejes de visión son casi paralelos. El grado de convergencia de los ojos permite al cerebro deducir la distancia a que se encuentra el objeto que en ese momento estamos mirando.

2.7.1.4. Acomodación

Cuando fijamos nuestra atención sobre un objeto, además de ajustar el ángulo de visión de los dos ojos se necesita variar la distancia focal del cristalino en función de la profundidad, con el fin de mantener enfocada la imagen. Al igual que sucedía con el fenómeno de la convergencia, del ajuste realizado para enfocar el objeto el cerebro puede extraer información sobre la distancia a que ese objeto se encuentra.

2.7.1.5. Limitaciones de los sistemas de RV

En los sistemas de RV resulta posible simular los dos primeros mecanismos de percepción de la profundidad. Así, las técnicas de estereoscopía permiten percibir en tres dimensiones incorporando en las imágenes el efecto de la disparidad binocular, mientras que los dispositivos de localización permiten simular el efecto del paralaje, modificando la imagen de acuerdo con la posición de la cabeza del usuario. El poder simular ambos mecanismos se debe a que los dos descansan, exclusivamente, en la posición y orientación relativas del usuario con respecto a los objetos, magnitudes éstas que un sistema de RV ya tiene en cuenta a la hora de realizar las proyecciones oportunas.

Sin embargo, los sistemas actuales no permiten simular ni el fenómeno de la de acomodación ni el de la convergencia, porque ambos mecanismos se basan en fenómenos internos del sistema de la visión humana. Para poder simular el fenómeno de la acomodación, por ejemplo, deberíamos ser capaces de detectar sobre qué objeto virtual está fijando su atención el usuario, y entonces presentar enfocado dicho objeto y desenfocados los demás. Hoy por hoy, es posible detectar la dirección en que un usuario mira, haciendo uso de las bandas electro-oculográficas de las que hablaremos en el capítulo 4. Sin embargo, no es posible aún discriminar con precisión a que objeto mira el usuario cuando hay varios objetos situados en una misma dirección (aproximada), pero a distintas profundidades. El hecho de no poder simular los fenómenos de la acomodación y convergencia no sólo hace que la sensación de relieve sea imperfecta, sino que causa algunos problemas de dolores de ojos en los sistemas estereoscópicos, especialmente debido a la acomodación.

2.7.2. Claves psicológicas

Las claves psicológicas de percepción de la profundidad son, por su parte, muchas y de muy diversa naturaleza, pero entre las principales podemos señalar los fenómenos de perspectiva lineal, interposición, sombreados y sombras, gradiente de textura y difuminación.

2.7.2.1. *Perspectiva lineal*

Es el efecto por el cual los objetos aparecen tanto más pequeños cuanto más lejos están. Conociendo el tamaño real de los objetos, el cerebro puede deducir la distancia a partir del tamaño aparente. Cuando no se conoce el tamaño real de los objetos, el cerebro puede aún inferir datos de distancias relativas entre objetos cuando hay varios similares en una misma escena, con diferentes tamaños aparentes.

2.7.2.2. *Interposición*

Es el fenómeno por el cual los objetos más próximos tapan a los más lejanos, lo que permite deducir cuál de ellos está situado más cerca.

2.7.2.3. *Sombreados y sombras*

Los sombreados son las diferencias en la intensidad de color a lo largo de la superficie de un objeto, debidas a la luz incidente. Así, las superficies que reciben la luz de frente aparecen más brillantes que las que reciben la luz de costado. Tanto los sombreados como las sombras que proyectan los objetos sobre el suelo (o sobre otros objetos) permiten al cerebro deducir la posición relativa de los objetos con respecto a las fuentes de luz y, por tanto, inferir datos de distancia.

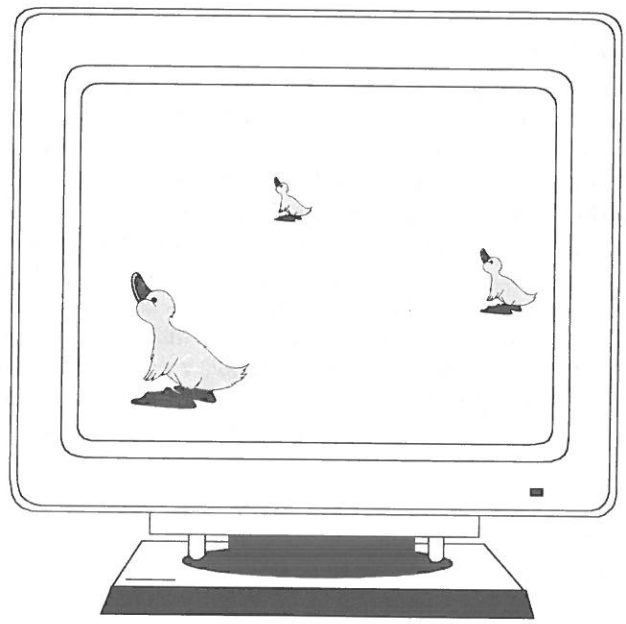

Figura 2.8. El fenómeno de la perspectiva lineal

2.7.2.4. *Gradiente de textura*

Es el efecto por el cual se pierde la capacidad de percibir los detalles de un objeto a medida que la distancia se incrementa. En consecuencia, el cerebro puede deducir información sobre la distancia de un objeto a partir del nivel de detalle percibido.

2.7.2.5. *Difuminación*

Es el efecto por el cual los colores se difuminan con la distancia, de forma que el cerebro extrae información sobre distancias a partir del brillo de los objetos.

Es importante observar que todos estos factores tienen su correspondiente "efecto óptico" asociado. Por ejemplo, puesto que los colores se difuminan con la distancia, el cerebro puede deducir la profundidad a que un objeto se encuentra a partir del color. Pero eso trae como resultado que, para dos objetos situados a la misma distancia, aquél que tenga colores más brillantes "parezca" más próximo que el otro.

Cualquier sistema de tratamiento de gráficos 3D genera de manera inmediata las claves de perspectiva, interposición y gradiente de textura al calcular la proyección de los objetos sobre la pantalla. El que se utilicen sombreados, sombras y difuminaciones, por el contrario, va a depender de la calidad que queramos dar a nuestras escenas, pudiendo o no implementarse dichos mecanismos a la hora de sintetizar las imágenes. Hablaremos más en detalle sobre estas cuestiones cuando tratemos el proceso de síntesis de las imágenes en el capítulo 5.

3
Dispositivos de salida

En el capítulo anterior hemos realizado un breve repaso de lo que los sistemas de RV son, y de cómo están constituidos. Ahora vamos a comenzar el análisis en detalle de los distintos componentes de un sistema de RV. Empezaremos por los dispositivos de salida, por cuanto son ellos quienes, en definitiva, van a determinar la visión que el usuario tenga del mundo virtual y, por consiguiente, del sistema.

El nombre genérico de dispositivos de salida abarca una serie de periféricos de muy diversa clase, que tienen por objeto proporcionar información al usuario acerca del estado del mundo virtual.

La clasificación más sencilla de estos dispositivos los agrupa según el sentido del usuario que pretendan estimular. Con este criterio, se clasifican en:

- Dispositivos de presentación, que tienen por objeto mostrar al usuario la información de carácter gráfico.
- Dispositivos de audio, encargados de generar la información sonora acerca del mundo virtual.
- Dispositivos de realimentación táctil, con los que se simulan las propiedades de los materiales que, como la rugosidad, el hombre percibe a través del sentido del tacto.
- Dispositivos de realimentación cinestésica, utilizados para simular propiedades mecánicas de los objetos, como la resistencia o la inercia.
- Dispositivos móviles, que permiten simular los movimientos de navegación a través del mundo virtual.

El papel fundamental corre, por supuesto, a cargo de los dispositivos de presentación, puesto que el hombre recibe la mayor parte de la información de su entorno a través del sentido de la vista. Serán estos dispositivos, junto con los de sonido, los que ocupen la mayor parte de nuestra exposición.

A lo largo del capítulo, explicaremos los principios en que se basa cada tipo de dispositivo, las tecnologías que utilizan y las tendencias que apuntan en el mundo comercial. Para cada categoría de dispositivo, señalaremos sus aplicaciones primordiales, los problemas que presenta su utilización y los principales productos comerciales existentes. En el Capítulo

8 tendremos ocasión de profundizar algo más en los avances relativos a los dispositivos de salida que cabe esperar en los próximos años.

3.1. DISPOSITIVOS DE PRESENTACIÓN

Los dispositivos de presentación tienen por objeto proporcionar al usuario la imagen del mundo virtual, es decir, información de carácter gráfico acerca de los objetos que componen el mundo virtual, sus posiciones y sus características.

Los sistemas de realidad virtual utilizan técnicas de tratamiento de gráficos 3D para generar la imagen a partir de una definición tridimensional del mundo virtual y de los objetos que lo forman. Dicha imagen será luego mostrada al usuario empleando un dispositivo de presentación adecuado.

Las características de la imagen van a venir determinadas tanto por la complejidad de los algoritmos de síntesis como por la calidad del dispositivo de presentación. Ambos aspectos tienen que estar equilibrados a la hora de diseñar un sistema de RV. De nada sirve implementar complejos algoritmos de iluminación de las escenas, si luego empleamos un dispositivo de baja resolución para presentar las imágenes. Como de nada sirve, tampoco, utilizar un dispositivo de alta resolución, si la potencia de proceso disponible no permite trabajar con objetos muy detallados.

Normalmente, los requerimientos de la aplicación van a determinar la calidad mínima de la imagen, y a partir de ahí deberemos dimensionar la capacidad de proceso del sistema, seleccionar los algoritmos de síntesis apropiados y escoger un dispositivo de presentación que esté en consonancia con lo anterior.

3.1.1. Tipos de dispositivos

Además de los dispositivos de carácter convencional (como monitores y pantallas de proyección) empleados en los sistemas proyectivos y de sobremesa, los sistemas de RV pueden utilizar tres tipos de dispositivos de presentación especiales:

- Visiocascos.
- Sistemas binoculares.
- Gafas estereoscópicas.

Tanto los visiocascos como los sistemas binoculares se emplean en aplicaciones inmersivas, donde el usuario ve únicamente las imágenes procedentes del mundo virtual, y no lo que hay a su alrededor. Ambos tipos de sistemas proporcionan siempre una visión estereoscópica al usuario, puesto que incorporan dos pantallas distintas, una por cada ojo.

Las gafas estereoscópicas, por su parte, se emplean en los sistemas inmersivos y proyectivos, precisamente para permitir al usuario la visión en relieve que un dispositivo de presentación convencional no le proporciona.

3.1.1.1. Sistemas inmersivos

Los visiocascos y sistemas binoculares (es decir, los dispositivos inmersivos) presentan la ventaja de que refuerzan en el usuario la sensación de realidad, al estar éste completamente circundado por el mundo virtual. A cambio, adolecen de tres problemas fundamentales.

El primero es que la inmersión dificulta las acciones del usuario, precisamente por impedirle la visión del mundo circundante. En particular, el usuario se ve imposibilitado para utilizar dispositivos físicos complejos, porque no puede verlos. Piénsese, por ejemplo, en lo que sucede en una aplicación donde el usuario necesite utilizar el teclado del ordenador para anotar cualquier tipo de información. Para este tipo de aplicaciones, obviamente, los sistemas de sobremesa o proyectivos con gafas estereoscópicas resultan más adecuados[1].

En ciertas aplicaciones, este problema se solventa parcialmente utilizando dispositivos semi-inmersivos, en los que se mezclan las imágenes sintéticas con las reales mediante un juego de espejos.

El segundo problema es que los dispositivos inmersivos necesitan algún mecanismo para conocer con cierta exactitud la posición y orientación del usuario, con vistas a mostrar el mundo virtual desde la perspectiva correcta. Este problema se resuelve empleando algún dispositivo de localización.

Sin embargo, las dos tecnologías de localización actualmente utilizadas en los sistemas inmersivos (de las que hablaremos en el próximo capítulo) tienen sus inconvenientes. Los dispositivos mecánicos utilizados en los sistemas binoculares, por ejemplo, permiten una localización bastante rápida y precisa, pero son muy engorrosos y tienen un alcance limitado. Los dispositivos electromagnéticos empleados en los visiocascos, por su parte, no son tan engorrosos, pero tienen también un alcance limitado, son más imprecisos, y también menos rápidos en determinar la posición del usuario, ya que se tarda un cierto tiempo en efectuar las mediciones necesarias para la localización, presentando por tanto una gran latencia.

El último problema es de índole psicológica. Se trata del hecho de que la sensación de estar inmersos en el mundo virtual jamás es completa, porque el usuario no pierde nunca de vista que el mundo real está ahí, rodeándolo, y que en cualquier momento puede destrozarse la espinilla con una mesita o engancharse con el cable del visiocasco y caerse de bruces al suelo. Este problema se puede obviar sólo si el usuario se encuentra dentro de un entorno controlado, como por ejemplo una cabina de vehículo simulada.

Pero quizá estemos realizando una pintura demasiado negra de los sistema inmersivos. A pesar de todo, los sistemas inmersivos presentan varias ventajas sobre los otros tipos de sistemas.

La primera es que, como ya hemos dicho, refuerzan la sensación de realismo de la aplicación, al aislar al usuario del mundo exterior.

---

1 Así, cuando el usuario deba emplear algún instrumento de control sofisticado, como sucede en ciertas aplicaciones de tele-robótica, se hace necesario reproducir en la pantalla un modelo del dispositivo (o sustituir el dispositivo físico por un dispositivo virtual que aparezca en la pantalla, lo que requiere un cierto aprendizaje y, además, no es siempre posible).

La segunda es que los sistemas inmersivos conducen a una mucho mayor naturalidad de la interfaz, especialmente cuando se emplean también dispositivos de entrada como los electroguantes. Esta naturalidad hace que, en determinadas aplicaciones como son las tareas de búsqueda, la efectividad del usuario se multiplique por dos (utilizando dispositivos de presentación con resolución equivalente), según estudios empíricos realizados por Randy Pausch en la Universidad de Virginia. Aunque, en honor a la verdad, hay que decir que si no se emplean resoluciones equivalentes, es decir, si se toman los dispositivos inmersivos y no inmersivos de más alta calidad existentes en la actualidad, la efectividad de los usuarios es mucho mayor con estos últimos por el simple hecho de que su calidad es mucho mayor.

Finalmente, existen aplicaciones en las que el uso de un sistema inmersivo es imprescindible, como por ejemplo cuando se desea probar condiciones de visibilidad por parte del usuario. En el capítulo 1 hemos descrito la utilización de un sistema de RV por parte de la compañía Caterpillar, para verificar mediante la simulación la visibilidad que el usuario de un vehículo tiene en condiciones reales de funcionamiento. Este tipo de verificación no podría llevarse a cabo con otro tipo de sistema.

3.1.1.2. Sistemas proyectivos y de sobremesa

En los sistemas proyectivos y de sobremesa se utiliza tecnología convencional (monitores y sistemas de proyección) para presentar las imágenes sintetizadas por el sistema.

Los dispositivos convencionales presentan la ventaja de la excelente resolución y calidad de la imagen, y de que no dificultan el uso de interfaces físicas. A cambio, no son inmersivos, el ángulo de visión es pequeño y, sobre todo, la imagen no es estereoscópica. Este último inconveniente se puede salvar empleando unos dispositivos especiales, las gafas estereoscópicas.

Los sistemas de sobremesa son el polo opuesto de los sistemas inmersivos. En los sistemas de sobremesa, el usuario ve el mundo virtual como si estuviera contemplándolo a través de una ventana. Obviamente, el usuario no tiene sensación de inmersión física dentro del mundo virtual, pero puede interactuar con el mundo virtual y desplazarse por él. Los sistemas de sobremesa presentan la ventaja de su bajo coste (no requieren ningún periférico especializado) y de que no impiden la visión del mundo real circundante, lo que puede ser muy importante en caso de que el usuario deba emplear algún instrumento complejo o introducir información textual en el ordenador. En cuanto a la desventaja de la no inmersión, algunos videojuegos demuestran a la perfección cómo puede conseguirse una sensación psicológica de inmersión aun cuando no exista inmersión física real.

En un sistema de sobremesa, se puede utilizar cualquiera de los mecanismos de localización existentes para conocer la posición de la cabeza del usuario, y así actualizar las imágenes de la pantalla correspondientemente. Con ello se puede conseguir el mismo efecto que al mirar a través de una ventana real: inclinándonos hacia la izquierda, aparece el paisaje situado más allá del borde derecho de la ventana, y viceversa. Pero la utilidad de esta técnica es bastante limitada y, de todas formas, el usuario sigue teniendo que mirar siempre hacia la pantalla, por lo que su libertad de movimientos está restringida. En consecuencia, no

suelen utilizarse dispositivos de localización en las aplicaciones de sobremesa, porque lo que añaden a la aplicación no compensa, por regla general, su coste.

Los sistemas proyectivos, finalmente, combinan todas las ventajas de los sistemas de sobremesa, en cuanto a calidad de la imagen se refiere, con la posibilidad de proporcionar al usuario la sensación de inmersión en el mundo virtual. En los sistemas proyectivos, el problema de la inmersión se resuelve empleando cabinas apropiadas en las que el usuario se introduce. En cierto modo, es como si el usuario se sumergiera en el mundo virtual dentro de su vehículo. El problema del ángulo de visión se resuelve a menudo colocando unas pantallas de proyección a continuación de otras, y actualizando las correspondientes imágenes de manera simultánea.

El hecho de utilizar un dispositivo en forma de cabina permite que este tipo de sistemas sea utilizado por varios usuarios de manera simultánea. A menudo, en los sistemas de RV proyectivos destinados al mercado del ocio, esta posibilidad se emplea con el fin de optimizar el rendimiento comercial, tal y como sucede con los simuladores tipo Venturer.

3.1.1.3. Técnicas de estereoscopía

Existen dos técnicas básicas de estereoscopía. La primera consiste en utilizar dos pantallas distintas, en cada una de las cuales se muestra la imagen correspondiente a un ojo. Esta es la técnica empleada en los sistemas inmersivos. Dichas pantallas pueden ser dos tubos de rayos catódicos (como en los sistemas binoculares) o dos pantallas de cristal líquido (como en los visiocascos). Para conseguir que cada ojo perciba sólo la imagen correspondiente, los ojos quedan físicamente acoplados a su respectiva fuente de luz. Esta técnica se denomina **multiplexación espacial**.

En los sistemas de sobremesa y proyectivos, pueden emplearse también dos fuentes diferentes de video, es decir, dos monitores distintos. Cada monitor se equipa entonces con filtros de polarización, y el usuario emplea unas gafas polarizadas que dejan pasar hacia cada ojo sólo una de las dos imágenes.

Sin embargo, lo más normal en los sistemas de sobremesa es emplear la segunda de las técnicas de estereoscopía, la llamada **multiplexación temporal**. Esta técnica se basa en utilizar una única pantalla, en la que se van mostrando alternativamente las imágenes correspondientes a los dos ojos. Para conseguir que cada ojo perciba únicamente una de las dos imágenes se emplean asimismo unas gafas especiales, las gafas estereoscópicas, de las que más adelante hablaremos.

3.1.1.4. Problemas de los sistemas estereoscópicos

Como ya hemos visto en el capítulo anterior, la utilización de la estereoscopía es bastante común en los sistemas de RV. Las técnicas de estereoscopía presentan la ventaja de que refuerzan el realismo de la aplicación al permitir la visión en relieve. Si empleamos, además, algún dispositivo de localización en conjunción con la visión estereoscópica, podemos

simular el fenómeno del paralaje, con lo que la percepción de la profundidad por parte del usuario es todavía mejor.

Sin embargo, cualquier técnica de estereoscopía tiene dos limitaciones fundamentales.

La primera es que, como ya hemos mencionado en el capítulo anterior, las técnicas actualmente disponibles no permiten simular el efecto de la acomodación ni el de la convergencia, porque no es posible, hoy por hoy, detectar cuál es el objeto sobre el que el usuario está fijando la vista.

Como consecuencia, el efecto de profundidad es imperfecto y, si la aplicación exige que el usuario cambie frecuentemente su atención entre objetos situados a diferentes profundidades, pueden aparecer dolores de cabeza y ojos.

La segunda limitación es de carácter práctico. Puesto que la distancia interocular varía de usuario a usuario, es necesario realizar un ajuste de las imágenes para cada uno de los distintos usuarios de un sistema. Si no se realiza un ajuste, el usuario puede ver las imágenes dobles. Si el ajuste es malo, pueden aparecer dolores de ojos con el uso continuado.

3.1.2. Visiocascos

Los visiocascos son unos dispositivos en forma de casco o visera, que el usuario lleva mientras se encuentra dentro del mundo virtual y en los que se incorporan un par de pantallas de cristal líquido pasivas (LCDs) en color, que le permiten una visión estereoscópica de la escena. El dispositivo se complementa con unos auriculares estereofónicos para el sonido y un sensor de posición, adherido al casco, que permite conocer la ubicación y orientación de la cabeza del usuario.

El cometido del visiocasco es doble: presentar una visión estereoscópica (en relieve) al usuario, y proporcionarle la sensación de inmersión impidiéndole la visión del mundo exterior.

La sensación de relieve por estereoscopía se consigue, como ya hemos explicado, mostrando en cada una de las dos pantallas imágenes ligeramente distintas, que se corresponden con la imagen del mundo virtual que cada uno de los dos ojos percibiría.

En cuanto al segundo cometido, es esta posibilidad de proporcionar una completa sensación de inmersión la principal ventaja de los visiocascos, puesto que aumenta la apariencia de realidad de la aplicación al perder el usuario la referencia del mundo real.

Para que la sensación de inmersión sea completa, es necesario que el visiocasco incorpore un dispositivo de localización que permita determinar la orientación de la cabeza del usuario. De ese modo, el sistema puede calcular de modo continuo la perspectiva correcta del mundo virtual que hay que presentar al usuario.

3.1.2.1. Características técnicas

El motivo por el que se emplean pantallas LCD en los visiocascos es que éstas son ligeras y poco voluminosas, requisitos éstos indispensables desde el momento en que el usuario debe

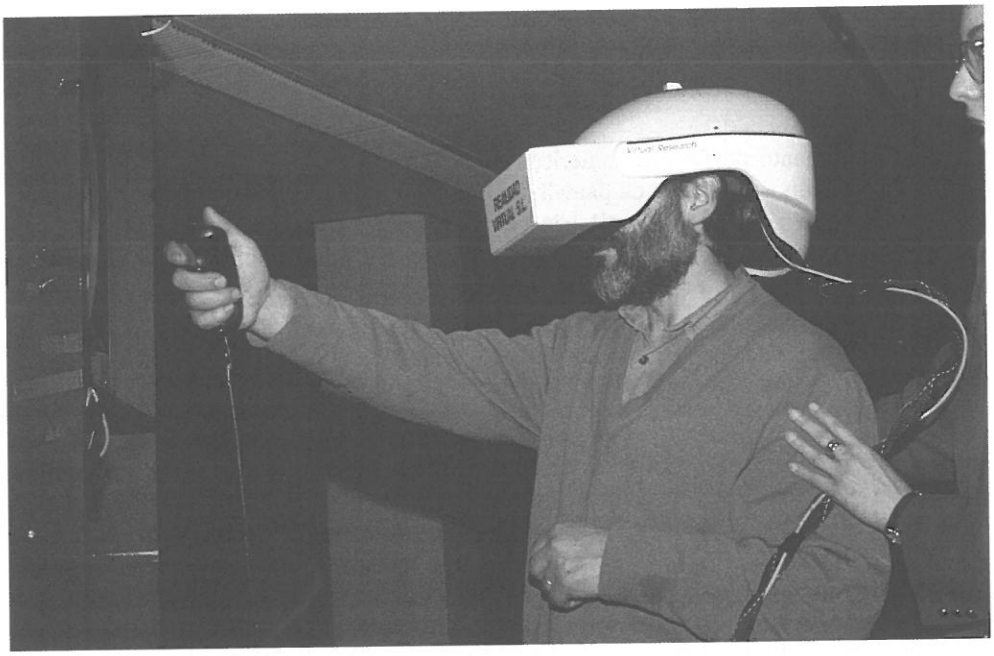

Figura 3.1. Visiocasco.
(Fotografía de Alejandro Sacristán)

llevar puesto el visiocasco en la cabeza. Las dos pantallas llevan sendas lentes de aumento, para agrandar el tamaño de la imagen, y el ángulo de visión que se consigue es bueno: hasta unos 110º en horizontal y 90º en vertical. Asimismo se suelen incorporar elementos de filtrado para reducir la sensación de granularidad que las pantallas LCD proporcionan.

A cambio de su poco peso y reducido volumen, los LCDs pasivos presentan la desventaja de su escasa resolución y de que la calidad de la imagen es baja, debido a los problemas de brillo y contraste que los LCDs presentan. La resolución máxima disponible actualmente es de unos 400x300 pixels por cada pantalla, aunque algunos modelos tienen una resolución cuatro veces más baja[1].

A la hora de comparar con la resolución de un monitor convencional hay que tener en cuenta que, en el visiocasco, ésta se divide entre un campo de visión mucho mayor, lo que se traduce en una resolución *relativa* todavía más baja, la cual es tan pobre que no resulta posible utilizar texto dentro de las aplicaciones de realidad virtual que emplean visiocasco.

1 En las pantallas LCD, una cosa es la resolución de la matriz LCD (es decir, el número de puntos que la componen), y otra la resolución en pixels. Cada pixel requiere tres puntos de la matriz, uno por color. Normalmente, los fabricantes suelen dar la resolución de la matriz, así que hay que dividir el número total de puntos por tres.

Existen visiocascos con LCDs monocromos que tienen una resolución mayor, con un número de pixels similar al de una pantalla VGA normal. Sin embargo, la carencia de color limita el número de aplicaciones y la resolución sigue siendo baja (teniendo en cuenta el ángulo de visión).

El Departamento de Defensa americano tiene en marcha una serie de proyectos que permitirán, en 3 años, disponer de pantallas LCD color de alta resolución para visiocascos. En Europa también se está desarrollando tecnología para la fabricación de visiocascos de alta resolución, dentro de un proyecto patrocinado por la ESA (European Space Agency) en el que participa la empresa española Construcciones Aeronáuticas.

Actualmente se está trabajando en el desarrollo de nuevas tecnologías, como la proyección retinal, en la que las imágenes se proyectan directamente sobre la retina mediante un rayo láser, en lugar de emplear LCDs. Pero estas tecnologías se encuentran aún en fase experimental.

Algunas empresas, como Kaiser Electro-Optics, Honeywell, CAE Electronics y Sextant Avionique, comercializan visiocascos "de transparencia", que permiten superponer una serie de imágenes gráficas sobre imágenes reales. Dichos visiocascos se utilizan en aviónica.

3.1.2.2. Problemas de los visiocascos

A pesar de sus innegables virtudes, los visiocascos presentan, hoy por hoy, una serie de problemas que limitan grandemente su campo de aplicación práctica. Algunos de dichos problemas son comunes a cualquier dispositivo inmersivo, como la dificultad de usar dispositivos complejos con el visiocasco puesto o la sensación de indefensión que el usuario experimenta al no poder ver nada de su entorno.

Otros problemas son específicos de los visiocascos. Ya hemos mencionado antes la baja calidad de imagen que se consigue, aunque es de esperar que este problema quede resuelto en un plazo de tiempo relativamente corto, a medida que la tecnología de pantallas LCD avance.

También hemos hablado anteriormente del problema de la latencia. Como resultado de la lentitud de medición de los dispositivos electromagnéticos, hay un cierto retardo desde el momento en que un usuario efectúa un movimiento hasta que la imagen comienza a cambiar en la pantalla, lo que puede provocar la aparición del mareo de simulación, al que aludimos en un capítulo anterior.

El último problema de los visiocascos también se asocia al mecanismo de localización utilizado, y se trata del hecho de que el control de movimiento de las imágenes es "difuso". En un sistema proyectivo o de sobremesa, el usuario controla a voluntad (mediante un joystick, un ratón o cualquier otro dispositivo) sus movimientos, pudiendo parar siempre que lo desee. En consecuencia, el sistema sabe en todo momento si el usuario desea o no moverse.

En los sistemas que utilizan visiocasco, sin embargo, la imagen depende no sólo de los controles manuales de movimiento de los que el usuario disponga, sino también de la posición de la cabeza de éste. Dado que resulta bastante complicado, para cualquier persona normal,

mantener la cabeza (o cualquier otra parte del cuerpo) en una posición perfectamente estática, el resultado es una especie de movimiento perpetuo de la imagen. Este efecto aumenta si el dispositivo de localización utilizado presenta un cierto grado de imprecisión.

El problema que este movimiento perpetuo conlleva es que, además de consumir una capacidad de proceso considerable en actualizaciones inútiles de la imagen, impide utilizar de forma conveniente ciertas técnicas utilizadas para mejorar el aspecto gráfico del mundo virtual (como la técnica de refinamientos sucesivos descrita en el capítulo 5). La única solución a este problema consiste en realizar un filtrado de la señal indicativa de la posición del usuario, lo que hace que se incremente aún más la latencia.

3.1.2.3. Sistemas comerciales existentes

La empresa californiana VPL Research fue la primera en comercializar un visiocasco, en 1989. En la actualidad fabrica dos modelos: el Eyephone LX y el Eyephone HRX. El modelo LX , con una resolución de 440x236 y un peso de 1,2 Kg, tiene un ángulo de visión de 108º en horizontal y 76º en vertical. Utiliza un sensor de posición Isotrak, de la empresa Polhemus, que se conecta al ordenador a través del puerto serie RS-232. El visiocasco tiene entrada de video NTSC. Su precio actual es de 11.000$. El HRX, más reciente, es de mayor resolución, con 720x480.

La empresa Virtual Research fabrica varios modelos de visiocasco, con distintas resoluciones:

- El Flight Helmet, con 360x240.
- El Eyegen 3, de 493x234.
- El VR4, de 742x230.

Entre los otros modelos de visiocascos existentes en el mercado podemos citar:

- CyberFace 2 de Leep Systems, que utiliza las mismas pantallas LCD que el Eyephone LX.
- Tier 1, de VRontier Worlds, con una resolución de 479x234 y un ángulo de visión de 112º.
- MRG 2, de Liquid Image Corp. Es un visiocasco monoscópico, con una resolución de 720x240 y un ángulo de visión de 100º, que admite señales NTSC y RGB.
- Visette, de la empresa Virtuality.
- PDS-Glasses, de Virtual I/O.

Recientemente han comenzado a hacer su aparición visiocascos de bajo coste, por debajo de los 1.000$, como el StuntMaster de VictorMaxx Amusements, o el 2001 Virtual Reality Video Visor, de la empresa Astounding Technologies. Este último incorpora un dispositivo de localización de Fakespace y auriculares estereofónicos, admite entradas NTSC y PAL, y tiene una resolución de 428x244, con un ángulo de visión de 60º. También la empresa Virtual I/O tiene previsto sacar en el primer trimestre de 1995 un nuevo casco, el PDS Gamer, cuyo precio estará en torno a los 250$.

Otras empresas, como la francesa Simulis, han anunciado el próximo lanzamiento de nuevos visiocascos, con resoluciones dos veces superiores a las actuales y un campo visual de 180º (utilizando cuatro pantallas LCD en lugar de dos).

3.1.3. Sistemas binoculares

Los sistemas binoculares emplean tubos de rayos catódicos (CRTs) en lugar de pantallas de cristal líquido. Los tubos de rayos catódicos permiten una resolución y calidad de la imagen mucho mayores que los LCDs, pero presentan la desventaja de ser excesivamente pesados y voluminosos como para que el usuario pueda llevarlos en la cabeza.

La solución empleada en los sistemas binoculares consiste en acoplar los tubos de rayos catódicos a un dispositivo mecánico con una serie de contrapesos. El usuario emplea el sistema como si se tratara de un periscopio, acercando los ojos al binocular cada vez que desea observar el mundo virtual. Para cambiar de orientación o de posición, el usuario empuja el binocular en la dirección adecuada. Los contrapesos del sistema mecánico permiten realizar dicha operación sin esfuerzo.

El dispositivo mecánico al que se acoplan los CRTS presenta una utilidad adicional, que es la de permitir medir la posición y orientación del usuario. Para ello, el dispositivo se

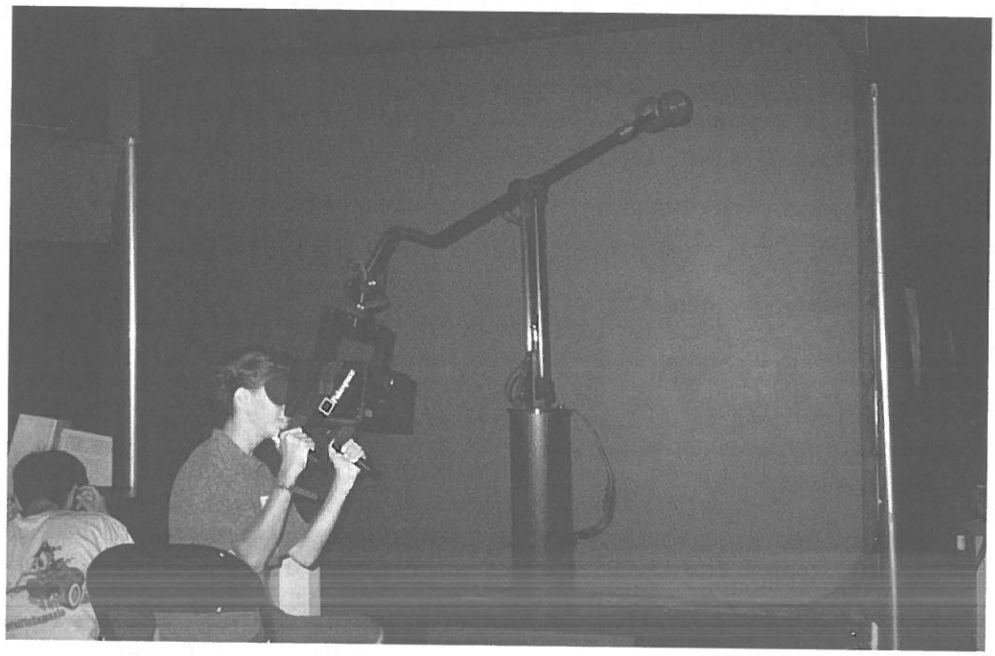

Figura 3.2. Sistema binocular
(Fotografía de Alberto Arnedillo)

fabrica mediante una serie de segmentos articulados (normalmente cinco o seis). Al moverse el usuario, los ángulos que forman los segmentos se modifican y unos sensores, que miden dichos ángulos, permiten determinar la posición y orientación de la cabeza del usuario.

3.1.3.1. Ventajas y desventajas de los sistemas binoculares

Los sistemas binoculares presentan mejor resolución y calidad de imagen que los visiocascos, debido a la tecnología utilizada, sin perder por ello ángulo de visión. También son menos problemáticos de cara a la utilización de instrumentos complejos por parte del usuario, y no presentan el problema de la "determinación difusa" de la posición: cuando el usuario quiere utilizar un cierto instrumento o desea fijar su orientación dentro del mundo virtual, no tiene más que soltar el periscopio, encargándose el sistema de contrapesos de mantener el binocular estable.

Las desventajas son el alto coste del dispositivo, su gran volumen, las limitaciones que el sistema mecánico impone a los movimientos de los usuarios, y el hecho de que el usuario tenga que utilizar las manos para moverlo. Como forma de paliar este último problema, se suelen incluir en los manillares del periscopio una serie de botones, con el fin de poder efectuar determinados controles sin soltar el aparato.

Otra desventaja es que, hasta hace poco tiempo, los CRTs empleados en los sistemas binoculares eran monocromos o disponían, como mucho, de dos canales de color, en lugar de los tres existentes en un monitor de sobremesa. Pero últimamente están empezando a salir al mercado los primeros modelos de CRTs para sistemas binoculares con capacidad completa de color.

3.1.3.2. Sistemas comerciales existentes

El más conocido de los sistemas binoculares es el BOOM, de la empresa FakeSpace Labs. Hay tres modelos: el BOOM II, el BOOM IIC y el BOOM 3C, que es el más reciente de todos ellos. El BOOM II permite una resolución máxima de 480x480, el BOOM IIC alcanza una resolución de 1000x1000, y el modelo 3C permite una resolución de 1280x960. Estas resoluciones son para el modo de trabajo entrelazado. En modo no entrelazado, la resolución es la mitad.

Recientemente, Tektronix ha desarrollado las pantallas Nucolor, que utilizan tubos de rayos catódicos de bajo peso y filtros LCD en color, de forma que pueden obtenerse imágenes color de alta resolución. La primera empresa en sacar al mercado un casco basado en esta tecnología ha sido Virtual I/O, que a mediados de 1994 presentó el casco Ultraview 2020, con una resolución de 640x480 (no entrelazado), y un ángulo de visión de 50º. El precio de estos cascos, sin embargo, es prohibitivo: de 20.000$ a 40.000$, según opciones..

3.1.4. Gafas estereoscópicas

Los sistemas proyectivos y de sobremesa suelen emplear la técnica de la multiplexación temporal para conseguir el efecto de estereoscopía. Esta técnica se basa, como ya hemos

comentado, en mostrar alternativamente, en la misma pantalla, las imágenes correspondientes a los dos ojos. El usuario utiliza unas gafas especiales, con el fin de conseguir que cada ojo perciba únicamente una de las dos imágenes.

Existen dos tipos de gafas estereoscópicas, las pasivas y las activas, siendo estas últimas las más utilizadas en la actualidad.

Las gafas estereoscópicas pasivas son gafas con polarización circular, donde cada cristal está polarizado en un sentido. Delante del monitor se coloca un filtro que se sincroniza con la señal de refresco de pantalla (la señal que indica que el sistema ha terminado de sintetizar la siguiente imagen). Cuando se muestra la primera imagen, el filtro la polariza en un sentido, de forma que la imagen sólo atravesará uno de los dos cristales de las gafas (el que esté polarizado en el mismo sentido). Al mostrarse la segunda imagen, el filtro conmuta y la polariza en sentido opuesto, con lo que la imagen atravesará el otro cristal de las gafas.

Las gafas estereoscópicas activas, por el contrario, tienen un par de cristales LCD y un dispositivo receptor que se sincroniza con la señal de refresco de pantalla. Ambos cristales LCD se vuelven opacos alternativamente, de acuerdo con las señales recibidas, de forma que se interrumpe la visión por el ojo correspondiente. Cuando en el monitor se está mostrando la imagen correspondiente al ojo izquierdo, las gafas impiden el paso de la luz hacia el ojo derecho, y viceversa.

Ambos sistemas permiten que varios usuarios contemplen simultáneamente la misma imagen tridimensional en la pantalla, por lo que pueden ser usados en presentaciones profesionales; sin embargo, cuando el número de espectadores es alto, la solución basada en filtros de polarización es más barata, porque se emplean gafas polarizadas normales.

La principal ventaja de las gafas estereoscópicas es que no alteran la forma en que se ven los objetos reales, por lo que el usuario puede emplear cualquier tipo de instrumento complejo para interactuar con la aplicación de realidad virtual. Además, pueden emplearse monitores de tecnología convencional, consiguiéndose una calidad de imagen muy superior a la de los sistemas inmersivos aunque, eso sí, con un ángulo de visión mucho menor.

3.1.4.1. Aliasing temporal

Cuando se emplean dos pantallas para lograr la visión estereoscópica, como sucede en los sistemas inmersivos, las imágenes que aparecen en las dos pantallas están sincronizadas, es decir, corresponden al mismo instante temporal. El sistema calcula las posiciones de los objetos y después muestra ambas imágenes simultáneamente.

Sin embargo, en los sistemas estereoscópicos con multiplexación temporal, caso de las gafas, las dos imágenes se muestran en distinto momento. Después de calcular las posiciones de los objetos, el sistema muestra primero una imagen y, después de un cierto tiempo, la segunda. Ello hace que aparezcan efectos ópticos indeseados, como una percepción incorrecta de la profundidad por parte del usuario, debido al retraso que una de las dos imágenes tiene con respecto a la otra. Este fenómeno se denomina **aliasing temporal**.

La única solución consiste en aumentar la velocidad de generación de las imágenes, porque el efecto óptico desaparece cuando el sistema entrega 60 imágenes por segundo a

cada ojo. Pero eso implica que la pantalla debe poder mostrar 120 imágenes por segundo, lo que hace necesario emplear un monitor especial.

3.1.4.2. Sistemas comerciales

Las gafas estereoscópicas más extendidas son las de la compañía californiana StereoGraphics, que comercializa las gafas CrystalEyes. CrystalEyes son unas gafas con lentes de cristal líquido que incorporan un receptor de rayos infrarrojos. Un dispositivo emisor conectado al ordenador permite sincronizar el receptor de infrarrojos de las gafas con el monitor. De esa forma, los dos cristales líquidos de las gafas se vuelven alternativamente opacos, dejando pasar sólo una de cada dos imágenes hacia cada ojo.

StereoGraphics comercializa otra versión, CrystalEyes-VR, que incorpora además un localizador ultrasónico para poder conocer la posición de la cabeza del usuario.

3.1.5. Sistemas autoestereoscópicos

En la actualidad existe un considerable interés en la investigación de los denominados sistemas autoesteroscópicos, es decir, sistemas que generan imágenes que pueden ser percibidas en relieve sin necesidad de enviar una imagen diferente a cada ojo.

Las dos tecnologías más estudiadas son:

- Los sistemas holográficos, que generan imágenes en tres dimensiones por interferencia de dos rayos láser.
- La tecnología de pantallas multiplanares, donde se utiliza un espejo giratorio para reflejar un rayo láser con el fin de representar en tres dimensiones una imagen descompuesta en múltiples planos 2D.

Entre los sistemas comerciales podemos citar el sistema CyberScope de la empresa americana 3-D ImageTek, y el monitor 3-D de la empresa noruega Stereovisjon.

3.2. DISPOSITIVOS DE AUDIO

3.2.1. El sonido en las aplicaciones de RV

El sonido tiene cuatro funciones primordiales en una aplicación de RV.

En primer lugar se encuentra lo que podríamos denominar **función informativa**, común a casi todos los tipos de aplicaciones. Las señales acústicas permiten informar al usuario de la ocurrencia de determinados eventos. Sirven, por ejemplo, para confirmarle que ha pulsado correctamente un botón, o para notificarle una condición de alarma o un cierto estado del sistema. En muchas ocasiones, el usuario necesita obtener una realimentación sobre las acciones que realiza, y el sonido es un medio conveniente de proporcionarle dicha información, sin necesidad de sobrecargar la pantalla con más información visual.

En segundo lugar está la **función metafórica**. El sonido, al igual que la imagen, puede emplearse para "traducir" una serie de datos a un formato fácilmente entendible por el

usuario. Este tipo de aplicación es bastante común en electromedicina, por ejemplo, donde se pueden traducir ciertas constantes vitales a sonidos en una unidad de cuidados intensivos. El sonido presenta la ventaja sobre la imagen, para este tipo de aplicación, de que no requiere la atención constante del usuario. Como ejemplo de función metafórica en un programa de RV, podemos pensar en una aplicación de visualización científica, donde se puede utilizar un determinado sonido para indicar al usuario el valor de una cierta magnitud (por ejemplo, un campo electromagnético), de forma que el sonido se va haciendo más o menos agudo a medida que el usuario se desplaza por el mundo virtual.

Estas dos funciones se pueden fácilmente incorporar a una aplicación utilizando mecanismos clásicos: tarjetas de sonido simples o, incluso, el propio altavoz del ordenador.

La tercera función es la **función artística**, donde el sonido se emplea en forma de música de fondo. Las secuencias musicales contribuyen a incrementar el atractivo de una aplicación, e incluso permiten influenciar, en cierta medida, el estado de ánimo del usuario, función ésta de la que se hace un amplio uso en los videojuegos.

La incorporación de secuencias musicales de una cierta calidad requiere utilizar dispositivos especializados, que pueden ser tan simples como una tarjeta de sonido convencional, o tan complejos como sintetizadores MIDI profesionales, dependiendo de qué calidad se desee obtener. Normalmente, las aplicaciones incorporan varios drivers, de forma que se pueden emplear con diversos tipos de dispositivo.

La última función es la que podemos denominar **función descriptiva**. Esta es la función realizada por todos los efectos de sonido que describen sucesos que tienen lugar dentro del mundo virtual. Los efectos de sonido, como el sonido del vidrio roto al caerse un vaso, el sonido del impacto cuando dos objetos chocan, etc... contribuyen al realismo de una escena casi tanto como la propia imagen. El poder escuchar lo que acontece dentro del mundo virtual no sólo contribuye a aumentar la sensación psicológica de inmersión y de "realidad", sino que permite al usuario comprender más fácilmente una escena.

Los efectos de sonido pueden generarse haciendo uso del mismo tipo de dispositivo que para las secuencias musicales. Pero, además, es posible incrementar el realismo de los efectos recurriendo a las llamadas técnicas de localización del sonido, que se explican con detalle en la sección dedicada al sonido 3-D.

El proceso de generación de sonido en una aplicación de RV constará de tres pasos:

1) Creación de la secuencia sonora.
2) Procesado de la señal y localización espacial
3) Reproducción del sonido resultante.

3.2.2. Creación de las secuencias sonoras

Existen dos formas principales de crear las secuencias sonoras:

- Por reproducción de una secuencia pregrabada.
- Utilizando un sintetizador.

3.2.2.1. Reproducción de secuencias pregrabadas

La técnica de reproducción a partir de una secuencia pregrabada es, con mucho, la más simple. Durante la fase de desarrollo de la aplicación, se editan (mediante un sintetizador, un editor de formas de onda o cualquier otra herramienta similar) todas las secuencias sonoras que se vayan a utilizar. Después, dichas secuencias sonoras se muestrean, y las muestras correspondientes a cada secuencia se almacenan en un fichero, con el formato que se desee. A la hora de ejecutar la aplicación, cuando el sistema decide que es necesario reproducir un sonido, no tiene más que acudir al correspondiente fichero para tener completa la secuencia sonora.

El precio que hay que pagar por esta simplicidad, claro está, es que sólo se puede generar un número limitado de sonidos (aquéllos precisamente que se hayan grabado de antemano) y la capacidad de almacenamiento requerida es grande (porque hay que guardar todas las muestras de la secuencia).

Suponiendo que no tuviéramos que hacer ningún procesamiento posterior (como las técnicas de localización 3D), lo único que habría que incluir en el sistema es algún mecanismo hardware o software para regenerar, a partir de las muestras, el sonido deseado.

3.2.2.2. Síntesis de sonido

Utilizar sonidos pregrabados es, en cierto modo, como añadir un tocadiscos al sistema: a la hora de generar un cierto sonido, el sistema busca el disco correspondiente y lo hace sonar. Evidentemente, el sistema no podrá generar ninguna secuencia cuyo disco no posea.

¿Qué sucede si en lugar de un tocadiscos empleamos un piano, y enseñamos al sistema a tocar? O, dicho en palabras técnicas, ¿por qué no utilizar un sintetizador de sonido en lugar de pregrabar las secuencias?

Un sintetizador de sonido es, en esencia, un dispositivo al que podemos ordenar, en cualquier momento, que genere una determinada nota con los parámetros que queramos (volumen, duración, instrumento que debe tocarla, etc...). Al recibir esa orden, el sintetizador genera el sonido correspondiente.

Con un dispositivo así, podemos tocar cualquier secuencia de sonido que deseemos, sin más que enviar al sintetizador la secuencia adecuada de órdenes de alto nivel.

¿Cómo se aplicaría esto en un sistema de RV? De dos formas diferentes. Por una parte, podemos (al igual que con los sonidos pregrabados) componer todas las secuencias sonoras al desarrollar la aplicación pero, en lugar de almacenar las muestras de los sonidos, guardaríamos la secuencia de órdenes de alto nivel. Para secuencias sonoras no excesivamente complejas, podemos conseguir con esto un ahorro importante de capacidad de almacenamiento.

Pero no tenemos por qué limitarnos sólo a reproducir secuencias previamente compuestas. Con un sintetizador, podemos generar cualquier secuencia que deseemos, en cualquier momento. Podemos pensar, por ejemplo, en el sonido de una gota de agua cayendo dentro

de un vaso virtual. El sintetizador le permite al sistema ir variando la frecuencia del sonido a medida que el vaso se llena, exactamente igual que sucede en la realidad.

Un ejemplo más complejo sería el de una aplicación capaz de mostrar en la pantalla un instrumento musical virtual que el usuario pudiera "tocar". Puesto que no conocemos de antemano la secuencia que el usuario interpretará, es imposible pregrabar todas las melodías. Pero un sintetizador tipo MIDI nos permite ir generando de forma muy sencilla las notas, a medida que el usuario las toca.

Evidentemente, para poder emplear técnicas de síntesis de sonido en una aplicación de RV, deberemos incorporar en el sistema un dispositivo apropiado, por ejemplo un sintetizador MIDI.

3.2.3. Sonido 3-D

Una vez creada la secuencia sonora (bien accediendo a un fichero donde las muestras han sido grabadas con antelación o utilizando un sintetizador), ésta estará lista para ser reproducida. En el caso, por ejemplo, de la música de fondo o de los avisos acústicos de error, no es necesario ningún procesamiento ulterior.

Sin embargo, el caso de los sonidos "descriptivos" es diferente. Cuando en una aplicación de realidad virtual (o en un videojuego) se emplean efectos sonoros, el usuario utiliza la visión para identificar el origen del sonido. Si ve un objeto caer al suelo al mismo tiempo que oye el sonido de un golpe, el usuario asociará el sonido con el suceso que acaba de contemplar.

Esta asociación psicológica es perfectamente válida para muchas aplicaciones, especialmente las de sobremesa, pero en los sistemas inmersivos o proyectivos, puede acentuarse más aún la sensación de realidad incorporando mecanismos que permitan al usuario identificar la procedencia del sonido a partir del sonido mismo. Son las llamadas **técnicas de localización del sonido**, que no sólo permiten asociar sonidos con imágenes, sino también hacer que el usuario se percate de la proximidad de objetos, o de la ocurrencia de eventos, que caen fuera de su actual campo de visión.

Imaginemos, por ejemplo, una aplicación inmersiva en la que el usuario se encuentra metido dentro de una habitación, de espaldas a la puerta. Supongamos que fuéramos capaces de reproducir el sonido de apertura de la puerta de forma que el usuario lo percibiera como viniendo "de atrás". Ese sonido permitiría al usuario saber que alguien está abriendo la puerta, aunque suceda a espaldas suyas, y podría girarse para ver qué sucede.

La más simple de las técnicas de localización del sonido es la de la audición estereofónica. Esta técnica, bastante familiar para todo aquél que tenga en casa un equipo de música, permite "situar" un sonido en una dirección determinada. Para ello se utilizan dos altavoces, por los que se emite el mismo sonido, pero a distinto volumen. La diferencia de volumen de las dos señales que llegan a nuestros oídos hacen que detectemos el sonido como procedente de un determinado punto, situado en la línea de unión de los dos altavoces.

Es evidente que, para poder utilizar esta técnica en un programa de RV, la aplicación deberá calcular en cada momento la posición relativa del usuario con respecto a la fuente de sonido, y controlar el volumen de los dos altavoces correspondientemente.

Sin embargo, esta técnica no tiene mucha utilidad en una aplicación inmersiva. La audición estéreo convencional sólo permite situar el punto de origen del sonido en la línea de unión de los altavoces, mientras que en un mundo virtual de tres dimensiones un sonido puede venir de cualquier parte del espacio. Se hace pues, necesario, emplear alguna técnica más sofisticada. Se puede, por ejemplo, generalizar la técnica del estéreo y añadir más altavoces (más fuentes de sonido), tal como sucede con los equipos de música con sonido cuadrafónico.

Pero la técnica que se suele emplear en la actualidad es otra, basada en una simulación del proceso real de la audición humana. En el mundo real, el ser humano es perfectamente capaz de determinar la procedencia del sonido a partir de su fuente, es decir, del objeto que lo genera. ¿De qué forma localiza el cerebro humano la procedencia de cada sonido?

El hombre es capaz de ver en tres dimensiones a partir de la comparación de las imágenes, ligeramente diferentes, que le llegan a los dos ojos. Con el sentido del oído sucede lo mismo: lo que percibimos por ambos oídos no es exactamente lo mismo, y son esas diferencias las que permiten al cerebro determinar la procedencia de un sonido.

Las diferencias entre lo que percibimos por uno y otro oído se deben a dos mecanismos distintos. En primer lugar, el sonido no llega a los dos oídos al mismo tiempo, sino que llegará primero al oído situado más cerca de la fuente de sonido. El retardo entre las dos señales (denominado retardo interauricular) es muy pequeño, pero suficiente como para poder extraer una primera información acerca de la posición relativa de la fuente con respecto a ambos oídos. Aún así, esta información no es todavía bastante, y aquí interviene el segundo de los mecanismos.

¿Se ha preguntado alguna vez para qué sirven los pliegues de la oreja? Pues esa forma peculiar de nuestro pabellón auditivo contribuye de manera decisiva a la tarea de localizar los sonidos. El sonido no es más que una vibración del aire, y el pabellón auditivo, con su forma peculiar, actúa como un filtro que distorsiona la señal sonora de una forma que depende de la dirección de la que el sonido provenga. De esa manera, se introduce otra diferencia más entre las señales que llegan a los dos oídos.

Estas dos informaciones combinadas son las que permiten al cerebro determinar, con una relativa precisión, en qué dirección del espacio de tres dimensiones está situada la fuente del sonido.

Pues bien, con las técnicas actuales de procesado digital de la señal resulta perfectamente posible coger un cierto sonido y procesarlo mediante un algoritmo que simule el **retardo interauricular** y el efecto de **filtrado del pabellón auditivo**. Basta entonces con hacer escuchar al usuario, mediante unos auriculares, las dos señales ligeramente diferentes así obtenidas, para que éste sea perfectamente capaz de determinar la dirección de la fuente virtual de sonido.

Esta técnica fue inventada en 1988 por Scott Foster, de Crystal River Engineering, compañía que comercializa el Convolvotron, una tarjeta doble para PC que permite añadir sonido tridimensional a cualquier aplicación.

Por cada fuente de sonido es necesario realizar todo el procesado digital de la señal correspondiente, por lo que con una tarjeta sólo puede simularse un número limitado de fuentes de sonido simultáneas. En concreto, el Convolvotron permite simular hasta 4 fuentes de sonido distintas en ambiente anecoico, es decir, sin tener en cuenta las reflexiones del sonido en los objetos. Si se quiere disponer de más fuentes virtuales de sonido simultáneas, es necesario emplear más tarjetas.

Alternativamente, el Convolvotron permite emplear una única fuente de sonido virtual e introducir el efecto de la reflexión del sonido sobre un máximo de seis superficies. La tarjeta permite también simular el efecto Doppler.

El funcionamiento de la tarjeta es sencillo. Una vez definida la posición del usuario y de las distintas fuentes de sonido en el espacio tridimensional, la tarjeta calcula la señal que llegará a cada oído desde cada una de las fuentes (introduciendo el retardo y la distorsión debida al pabellón auditivo). Una vez hechos los cálculos para todas las fuentes de sonido, se suman por separado las señales correspondientes a uno y otro oído, y el resultado se lleva a los auriculares.

El Convolvotron se puede programar mediante una librería de funciones C, y tiene varias entradas diferentes de sonido, para poder conectar distintos dispositivos. Las señales de entrada se muestrean a una frecuencia de 50 KHz, mientras que la frecuencia de actualización de la posición de las fuentes de sonido virtual es de 33 Hz.

Crystal River Engineering comercializa también otra tarjeta para PC, denominada Beachtron, que permite simular dos fuentes diferentes de sonido y, además, proporciona los servicios de una tarjeta de sonido convencional. De hecho, la tarjeta Beachtron está basada en la tarjeta de sonido Multisound, de la compañía Turtle Beach Systems.

La compañía Focal Point 3D Audio fabrica otro sistema que opera basándose en un principio similar. La compañía Advanced Gravis ha licenciado la tecnología de Focal Point para su tarjeta de PC Gravis Ultrasound.

3.2.4. Reproducción del sonido

Una vez efectuado el procesamiento necesario, todas las secuencias de sonido (secuencias musicales, avisos acústicos, efectos sonoros) pueden sumarse y enviarse al dispositivo encargado de la reproducción. El diagrama del proceso global de generación del sonido sería el que muestra la figura 3.3.

Como vemos, los efectos sonoros se procesan para introducir el efecto de la localización 3D, obteniéndose dos señales. A cada una de estas dos señales se le suman las correspondientes a las secuencias musicales y avisos acústicos (que no necesitan procesamiento 3D), y el resultado se envía a los altavoces. Las señales correspondientes a las secuencias musicales

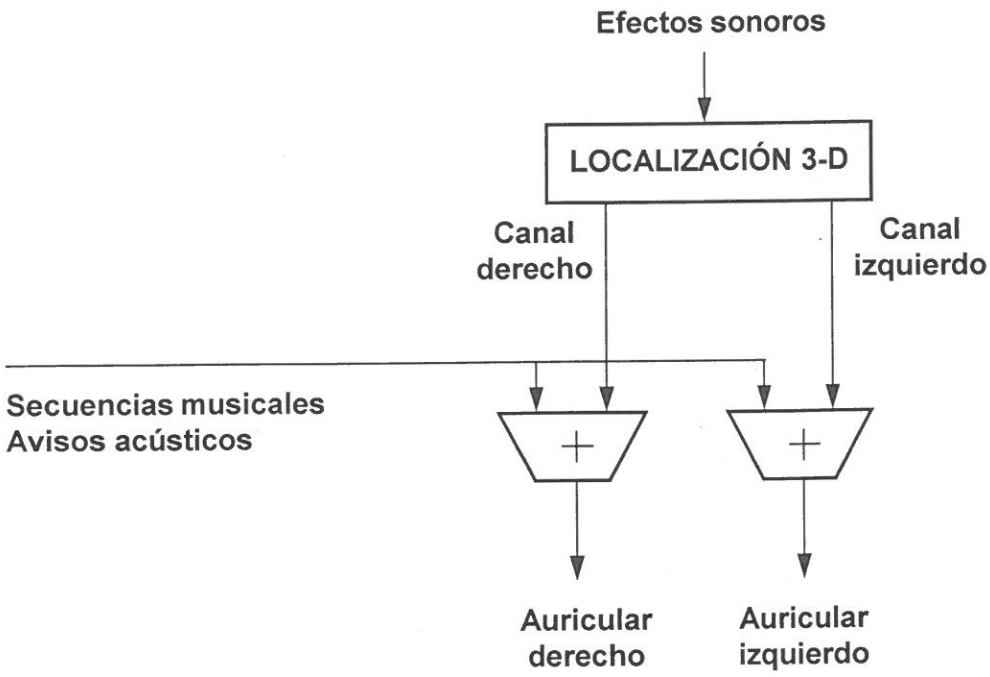

Figura 3.3. Tratamiento del sonido en una aplicación de RV

y avisos acústicos pueden ser las mismas para los dos canales (como en la figura), o diferentes (si se utiliza sonido estereofónico).

Cuando no sea necesario utilizar técnicas de localización de sonido, cualquier tarjeta de sonido convencional permite reproducir tanto secuencias pregrabadas como secuencias MIDI. La salida de la tarjeta se lleva directamente a los altavoces.

En el caso de utilizar técnicas de localización se puede emplear una tarjeta convencional, y llevar la salida a una tarjeta de sonido 3-D, que se encargará de efectuar el procesamiento requerido. Los altavoces se conectan a esta tarjeta de sonido 3-D. Alternativamente, puede emplearse una tarjeta como la Beachtron, que dispone de las dos funcionalidades (reproducción de secuencias y localización 3-D).

3.2.5. Sintetizadores de voz

Entre los dispositivos de sonido a veces utilizados en aplicaciones de RV se encuentran también los sintetizadores de voz. Los sintetizadores de voz permiten ofrecer información al usuario sobre variables de estado o sobre medidas realizadas, sin necesidad de saturar el

dispositivo de presentación gráfica. En este sentido, los sintetizadores realizan una función de carácter informativo acerca del estado del mundo virtual.

Los sintetizadores también permiten simular figuras con habla, pero hay que recalcar que la síntesis de voz sólo es necesaria cuando el mensaje que se va a transmitir cambia. Para mensajes fijos es mucho más eficiente (como es común en muchos videojuegos) emplear sonidos pregrabados.

3.3. OTROS DISPOSITIVOS DE SALIDA

Los dispositivos de presentación y de audio no son los únicos mediante los que un sistema de RV puede entregar información al usuario sobre el mundo virtual, aunque sí los más importantes.

Existen otros dispositivos, más especializados, que pueden completar la información visual y auditiva, para aumentar el realismo de las aplicaciones o para mejorar la naturalidad de la interfaz con el ordenador.

3.3.1. Plataformas móviles

Ya hemos comentado en el capítulo anterior los problemas de mareo que la navegación a través del mundo virtual acarrea, debido a la disparidad entre la información visual y lo que nuestro sentido del equilibrio detecta.

Los dispositivos móviles tienen por objeto, precisamente, simular los movimientos de navegación, no sólo para conseguir un mayor realismo, sino para compensar en parte las causas productoras del mareo.

En un sistema proyectivo, por ejemplo, podemos disponer la cabina encima de una plataforma móvil controlable desde un ordenador. Dependiendo de los movimientos que los usuarios realicen, el sistema hará inclinarse la plataforma a un lado y otro.

Este tipo de dispositivo no es especialmente exótico. Cualquiera que haya montado en un simulador tipo Venturer ha tenido la oportunidad de comprobar hasta qué punto aumenta la ilusión de inmersión cuando los movimientos de una plataforma se sincronizan con las correspondientes imágenes.

Pero el uso de este tipo de dispositivo no está restringido a los sistemas proyectivos. También en los sistemas inmersivos puede emplearse esta técnica con muy buen resultado para simular la navegación. La película "El cortador de césped" mostraba una de estas plataformas, donde los usuarios se colocaban en una especie de camilla provista de un visiocasco y unos mandos de control. La plataforma de la película no era real, pero el concepto es fácilmente implementable.

Las plataformas móviles disponen de tres ejes de giro. Normalmente, uno de los ejes regula el grado de inclinación hacia adelante o atrás, el otro regula el grado de inclinación a derecha o izquierda, y el tercero regula el *giro* a derecha o izquierda. La combinación de estos tres movimientos permite obtener cualquier inclinación (dentro, por supuesto, de los límites máximos de giro de cada eje).

El control de la plataforma se realiza desde un ordenador, a través de una unidad de control que transforma las órdenes del sistema en tensiones eléctricas. Dichas tensiones eléctricas hacen que unos pistones hidráulicos o neumáticos se extiendan o se contraigan, con lo que el ángulo del respectivo eje varía.

Dependiendo de la construcción mecánica de las plataformas, es posible que el giro alrededor de los distintos ejes esté limitado. Normalmente, en los sistemas proyectivos, con múltiples usuarios, los dispositivos hidráulicos o neumáticos se sitúan debajo de la plataforma, y la inclinación de ésta no puede sobrepasar unos límites prefijados. En estos casos no es posible, por ejemplo, poner la plataforma cabeza abajo. Esto no tiene necesariamente que ser un problema, por ejemplo si el usuario no puede llegar a estar situado cabeza abajo en el mundo virtual. Sin embargo, cuando el movimiento del usuario dentro del mundo virtual no está restringido, la simulación del mismo mediante este tipo de plataforma será imperfecta.

Hay otros dispositivos, como el Cybertron, en los que la plataforma está montada en un sistema de tres círculos cada uno de los cuales puede girar alrededor de su eje. En este caso, no existe limitación de giro por lo que se puede simular cualquier secuencia de navegación. Generalmente, estos dispositivos se emplean en aplicaciones inmersivas monousuario.

Es importante observar que las plataformas móviles no simulan las traslaciones a través del mundo virtual, sino las posiciones que el usuario adopta. Supongamos un usuario que está "volando" a ras del suelo dentro del mundo virtual. Mientras que no efectúe ninguna maniobra, la plataforma permanecerá quieta. Es en el momento en que el usuario de la orden de ascender cuando la plataforma se inclinará hacia atrás, para reflejar la nueva posición que el usuario tiene. Lo mismo sucederá cuando el usuario cambie su rumbo hacia la derecha o hacia la izquierda, o cuando descienda.

Pero existe otra posibilidad adicional, que se basa en el hecho de que es perfectamente posible simular la inercia de un cuerpo a través de su peso. ¿Qué sucede si, en nuestro ejemplo anterior, el usuario no efectúa un cambio de rumbo, sino una aceleración? Pues que podemos también inclinar hacia atrás la plataforma. Como la imagen le indica al usuario que continúa volando paralelo al suelo, su cerebro interpreta que esa fuerza que tira de él hacia atrás se debe a la aceleración, no a que ahora esté en una posición inclinada. De la misma manera, inclinando la plataforma hacia adelante podemos simular el efecto del frenado, e inclinándola hacia la derecha o la izquierda podemos simular la fuerza centrífuga asociada a un giro.

Entre las empresas que fabrican plataformas que pueden emplearse para aplicaciones de RV podemos citar las inglesas Denne Developments, Electrocoin y Air Muscle, así como las americanas Cynapse y Doron Precision.

3.3.2. Dispositivos táctiles

De los restantes sentidos a través de los cuales el hombre adquiere información, el único que se utiliza en las aplicaciones de realidad virtual actuales es el del tacto.

Si los ojos del usuario juegan un papel fundamental en las aplicaciones de RV como órgano por el que el sistema entrega información al usuario, las manos juegan otro papel no

menos importante, pero en sentido inverso. Es a través de las manos, fundamentalmente, como el usuario interactúa con el sistema, transmitiendo sus órdenes o entregando información. Casi todos los dispositivos que el usuario emplea para el control de las aplicaciones se manejan con la mano. Tan sólo escapan a esta regla los sistemas de reconocimiento de voz y los dispositivos de localización.

En la vida real sucede exactamente lo mismo. Es a través de las manos como controlamos la mayoría de los instrumentos y como empleamos la mayoría de los objetos. Es natural, por tanto, que los sistemas de RV utilicen ampliamente las manos para las labores de control.

En el otro sentido, es decir, para la información que fluye desde el sistema hacia el usuario, las manos juegan un papel menos trascendental (también como en el mundo real). La cantidad de información que el hombre adquiere a través del sentido del tacto es casi despreciable, si la comparamos con la que adquiere a través de la vista o el oído.

En determinadas circunstancias, sin embargo, la información que se adquiere a través de las manos puede tener una gran importancia en las aplicaciones de RV. Esta información puede clasificarse en dos diferentes tipos: información táctil e información cinestésica.

3.3.2.1. Dispositivos de realimentación táctil

La información táctil es aquélla que nos indica el contacto con un objeto, y que nos permite averiguar determinadas características del mismo, como la rugosidad de la superficie o su temperatura, a través de los sensores que el ser humano posee en la piel.

En una aplicación de realidad virtual, los dispositivos de realimentación táctil permiten, por ejemplo, que el usuario sienta el contacto con un objeto virtual cercano, contribuyendo así al realismo de la aplicación. También sirven como indicación para las operaciones que el usuario debe realizar: por ejemplo, si el usuario debe coger un objeto dentro del mundo virtual, se puede emplear un dispositivo de realimentación táctil como una forma natural de indicarle cuándo su mano está lo suficientemente cerca del objeto como para poder cogerlo.

Los dispositivos de realimentación táctil encuentran su aplicación fundamental en los sistemas inmersivos y en determinadas aplicaciones de tele-robótica.

Entre los dispositivos comerciales de realimentación táctil podemos citar el sistema TiNi Alloy, comercializado por la empresa americana Mondo-tronics. Este sistema utiliza unos botones de 20x9 mm que hacen presión sobre la piel bajo comando de una unidad de control, que se conecta al ordenador a través del puerto serie. Los botones pueden montarse, por ejemplo, en un electroguante o en un guante convencional.

Airmuscle Ltd comercializa otro sistema de realimentación táctil que emplea unas pequeñas almohadillas inflables para simular la presión sobre los objetos.

Los sistemas de realimentación táctil actuales adolecen de dos defectos fundamentales. El primero es su gran tamaño y baja frecuencia de activación, que impiden simular rugosidades "finas" de las superficies. Esta es una limitación de carácter tecnológico que los próximos años se encargarán de hacer desaparecer.

La segunda limitación es de carácter conceptual. Cuando tocamos un objeto en el mundo real, la sensación táctil (sensación de contacto físico) está complementada por una realimentación de carácter cinestésico: nuestros dedos no sólo sienten el contacto, sino que además no podemos continuar moviéndolos porque la superficie del objeto nos lo impide. En una escena virtual, por el contrario, la sensación táctil se presenta sola: aun cuando notemos (mediante los dispositivos tactores) el contacto con una superficie, esa superficie virtual no ofrece ninguna resistencia al avance de nuestros dedos, con lo que la sensación es algo irreal.

Por lo que a otras propiedades de los materiales se refiere, al margen del mero contacto físico, existen también empresas que comercializan componentes para la simulación de temperatura. La empresa C/M Research, por ejemplo, comercializa el sistema DTSS X10, que consta de 8 dispositivos termoeléctricos, que pueden acoplarse a los dedos para simular, bajo control de un ordenador, la temperatura de los objetos virtuales.

3.3.2.2. Dispositivos de realimentación cinestésica

Los dispositivos de realimentación cinestésica son los encargados de proporcionarnos información de carácter mecánico sobre la resistencia de los objetos.

Quizá el ejemplo más evidente sea el de los mandos de un avión. Aun cuando los sistemas electrónicos permitirían que el piloto pudiera accionar todos los mandos con un esfuerzo mínimo, los mandos de los aviones simulan una cierta resistencia, precisamente para ayudar al piloto a controlar mejor la maniobra.

Estos sistemas se emplean fundamentalmente en aplicaciones de simulación y de tele-robótica. En este último campo, por ejemplo, el añadir la realimentación cinestésica incrementa grandemente la efectividad de los operadores, al permitirles detectar de forma directa las colisiones del brazo articulado de un robot con alguna superficie, y al permitir simular propiedades de los materiales como la viscosidad.

4
Dispositivos de entrada

Los dispositivos de entrada tienen un doble cometido en los sistemas de RV:

Por un lado, permiten al usuario transmitir sus órdenes al sistema. Utilizando ciertos dispositivos de control, el usuario puede, por ejemplo, informar al sistema de que desea desplazarse o efectuar una determinada interacción con los objetos de un mundo virtual. Este tipo de dispositivos de entrada permite controlar de manera voluntaria y explícita el funcionamiento del sistema.

Pero el sistema también puede depender de otro tipo de controles, no explícitos. En una aplicación inmersiva, por ejemplo, el sistema deberá medir cuál es la dirección en la que el usuario está mirando, con el fin de presentar el mundo virtual desde la perspectiva correcta. Y en una aplicación de visualización científica, el sistema puede tener que adquirir los datos que luego va a mostrar al usuario. En estos casos, no es éste quien transmite órdenes explícitamente al sistema, sino que es el propio sistema quien determina, utilizando un dispositivo de medida, una serie de valores que le son necesarios.

En un sistema de RV puede emplearse cualquier dispositivo de medida o de control de los comúnmente empleados en otros tipos de aplicaciones. Pero los sistemas de RV presentan una serie de características distintivas que han obligado al desarrollo de nuevas tecnologías, específicamente pensadas para este tipo de aplicación:

- En primer lugar, los sistemas de RV plantean el problema de la localización del usuario. Los sistemas de RV necesitan conocer la posición y (en muchas ocasiones) la orientación del usuario, con vistas a generar las representaciones visuales y auditivas correctas. Este tipo de necesidad ha conducido al desarrollo de los dispositivos de localización, prácticamente privativos de los sistemas de RV, y a quienes, por su importancia, dedicamos un apartado completo del presente capítulo.
- En segundo lugar, el carácter tridimensional de las aplicaciones plantea el problema de la navegación del espacio virtual. Los dispositivos convencionales, como ratones o joysticks, fueron desarrollados pensando en entornos puramente bidimensionales, en los que la pantalla es casi como una hoja de papel en la que la información se muestra.

En aquellas aplicaciones que incorporaban ciertas características tridimensionales, como algunos juegos, se limitaban las posibilidades de navegación para poder seguir utilizando los controles bidimensionales. El advenimiento de aplicaciones tridimensionales puras, como son las de RV, obligó al desarrollo de nuevos modelos de ratones y joysticks pensados específicamente para la navegación en espacios de tres dimensiones.

- Finalmente, la necesidad de acentuar la apariencia de realidad del mundo virtual ha llevado al desarrollo de nuevas interfaces, más naturales, basadas en el uso de electroguantes y sistemas de reconocimiento de voz. La mano y la palabra son, en el mundo real, nuestros dos mecanismos principales de control de los objetos.

En este capítulo, vamos a pasar revista a los principales tipos de dispositivo de entrada que hoy en día se utilizan en los sistemas de realidad virtual, comenzando por los más importantes y característicos de todos ellos: los dispositivos de localización.

4.1. DISPOSITIVOS DE LOCALIZACIÓN

4.1.1. Funciones

Los dispositivos de localización sirven para medir la posición y orientación de cualquier objeto o parte del mismo en el espacio de tres dimensiones, necesidad ésta que surge a menudo en los sistemas de realidad virtual.

Existen cuatro aplicaciones básicas para este tipo de dispositivo, en un sistema de RV:

4.1.1.1. Detección de la orientación del usuario

En todo sistema de realidad virtual, éste necesita conocer la posición y orientación del usuario dentro del mundo virtual, con el fin de sintetizar las imágenes desde el adecuado punto de vista.

Consideremos primero el problema de la orientación. Vamos a suponer que el usuario está situado en una determinada posición del mundo virtual, y que lo único que está a su alcance es cambiar su orientación, es decir, cambiar la dirección en la cual está mirando.

En los sistemas inmersivos, el usuario emplea un visiocasco o un sistema binocular para conseguir la sensación de estar situado dentro del mundo virtual. En consecuencia, el control de la orientación debe ser automático: si el usuario vuelve la cabeza a un lado, su orientación dentro del mundo virtual debe cambiar correspondientemente.

Esto implica que debemos emplear algún mecanismo mediante el cual pueda el sistema detectar de manera automática en qué dirección está mirando el usuario. Dichos mecanismos son los dispositivos de localización.

En los sistemas no inmersivos, el usuario no está "dentro" del mundo virtual, sino que lo contempla a través de una "ventana", que es la pantalla del ordenador o del proyector. En estos sistemas, el cambio de orientación dentro del mundo virtual es un cambio de la

orientación *de la ventana*, no de la cabeza del usuario. En realidad, para poder ver a través de la ventana, el usuario debe estar mirando en la misma dirección, es decir, hacia la pantalla.

En estos sistemas, por consiguiente, no importa en absoluto saber a dónde dirige su mirada el usuario, por lo que no es necesario medir la orientación con un dispositivo de localización.

4.1.1.2. Detección de la posición del usuario

Los dispositivos de localización permiten, además, medir posiciones absolutas, y no sólo orientaciones.

En un sistema inmersivo, por ejemplo, se puede emplear el dispositivo de localización para conocer los desplazamientos que el usuario realiza, y actualizar su posición dentro del mundo virtual de acuerdo con ellos.

El problema estriba en que los dispositivos de localización actualmente existentes tienen un alcance bastante limitado (un par de metros). Eso quiere decir que, si no proporcionamos otro mecanismo de navegación, el usuario tendría unos movimientos bastante restringidos dentro del mundo virtual. Lo normal es, por tanto, dar al usuario algún otro mecanismo de navegación adicional, por ejemplo un joystick.

La mayoría de los sistemas inmersivos, por tanto, disponen de un dispositivo de localización mediante el que se detecta el punto de vista del usuario, y con el cual éste puede realizar pequeños desplazamientos dentro del mundo virtual. Para desplazamientos más grandes, el usuario emplea otro dispositivo de control.

En los sistemas no inmersivos, las tareas de navegación no pueden confiarse a un dispositivo de localización. En un sistema de sobremesa, el usuario está sentado en su silla, y no realiza desplazamiento alguno; mientras que en un sistema proyectivo, es la cabina la que supuestamente se desplaza por el mundo virtual, no teniendo importancia alguna los desplazamientos que el usuario pueda realizar dentro de la cabina.

En consecuencia, en los sistemas no inmersivos, el usuario empleará cualquier dispositivo de control (ratón 3-D, joystick, teclado, ...) para navegar a través del mundo virtual, no siendo estrictamente necesario un dispositivo de localización.

El único caso en que puede necesitarse un dispositivo de localización en un sistema no inmersivo es para simular el efecto del paralaje. Cuando, en el mundo real, miramos de frente a una ventana la imagen no es absolutamente estática: si nos inclinamos un poco a la derecha, podemos ver aparecer objetos por el borde izquierdo de la ventana, y viceversa. La única manera de simular este efecto es conocer, empleando un dispositivo de localización, la posición relativa del usuario con respecto a la pantalla.

4.1.1.3. Control de la aplicación

En los dos usos anteriormente mencionados, los dispositivos de localización sirven para el control *implícito* del funcionamiento del sistema. El usuario no transmite órdenes a través

del dispositivo, sino que es el sistema quien usa el dispositivo para averiguar dónde se encuentra el usuario situado.

Pero los dispositivos de localización también pueden emplearse para el control explícito de la aplicación de RV por parte del usuario, de dos maneras distintas.

En primer lugar, estos dispositivos pueden emplearse en conjunción con otras partes del cuerpo del usuario, no sólo la cabeza. En determinados sistemas, por ejemplo, podría emplear la mano para indicar en qué dirección desea desplazarse, pudiendo utilizarse un dispositivo de localización adherido a la mano para que el sistema sea capaz de detectar la dirección en la que apunta. Algunos modelos de electroguantes, de los que hablaremos en el siguiente apartado, incorporan dispositivos de localización con este propósito.

También pueden emplearse los dispositivos de localización para construir dispositivos de control genéricos. Es el caso de los ratones 3-D, que pueden formarse añadiendo los pulsadores necesarios a un dispositivo cualquiera de localización.

4.1.1.4. Digitalización de objetos

Finalmente, puesto que los dispositivos de localización pueden emplearse para determinar posiciones en el espacio de tres dimensiones, nada más natural que emplearlos para poder medir las dimensiones y la forma de los objetos.

Si queremos crear un modelo por ordenador de un objeto real, una forma simple de hacerlo es emplear un dispositivo de localización, con el que vamos midiendo en diversos puntos de la superficie. Las medidas así obtenidas (es decir, las coordenadas de los puntos) son introducidas de manera automática en el ordenador, quedando el modelo listo para su manipulación con cualquier programa de diseño comercial. Los equipos que permiten realizar dicha serie de medidas se denominan digitalizadores, y hablaremos de ellos en el capítulo dedicado al desarrollo de aplicaciones.

En esta sección vamos a pasar revista a las distintas técnicas existentes de localización: electromagnética, mecánica y ultrasónica.

4.1.2. Dispositivos electromagnéticos

¿Quién no conoce el procedimiento de la triangulación, por el cual resulta posible determinar las coordenadas de un objeto sabiendo su distancia a tres puntos de referencia?

Los dispositivos de localización electromagnéticos utilizan un procedimiento semejante al de la triangulación para determinar la posición y orientación de los objetos pero, en lugar de emplear tres puntos de referencia, se utiliza un emisor de señales radioeléctricas que emite, a intervalos regulares, tres señales electromagnéticas cuyos campos son ortogonales (perpendiculares) entre sí. Dichas señales se recogen en un receptor formado por tres bobinas perpendiculares, y se comparan. De esa comparación se deducen la posición y orientación que los receptores tienen con respecto al emisor.

Tanto el emisor como el receptor, o receptores, se conectan mediante un cable a un equipo electrónico, en el que se suelen encontrar los circuitos de generación del campo

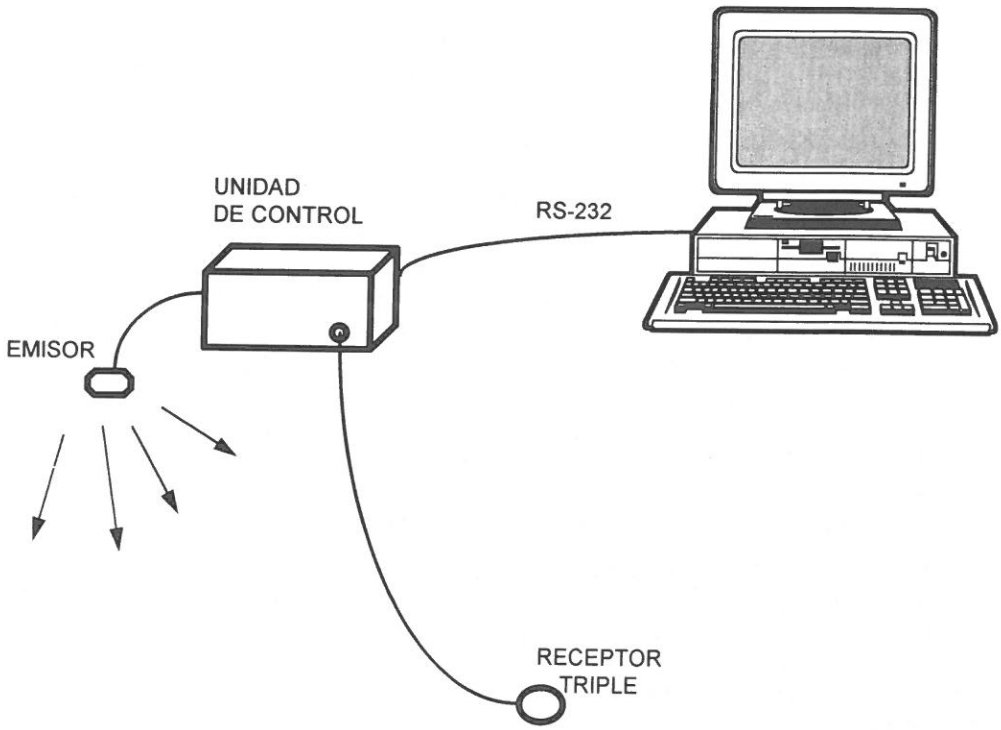

Figura 4.1. Localización electromagnética

magnético, así como un procesador digital de señal, para realizar los cálculos de la posición y orientación. Aunque dicho equipo suele ser una unidad autónoma separada del ordenador, funciona bajo su control y suele conectarse a través del puerto serie. Últimamente, sin embargo, han comenzado a aparecer equipos en tarjeta, que se insertan directamente en una de las ranuras de expansión de un PC.

La aplicación de realidad virtual puede interrogar cuando es necesario al dispositivo de localización, y éste le devuelve la posición y orientación de la cabeza del usuario. De esa manera, el programa de simulación sabe cuál es el punto de vista actual que debe utilizar para dibujar la escena.

Los dispositivos de localización electromagnéticos presentan una serie de problemas inherentes a la tecnología empleada:

- En primer lugar, el alcance del sistema está limitado por la potencia del emisor y porque el error aumenta con la distancia a que los receptores se encuentran. El error contenido en las medidas generadas por estos dispositivos suele ser bastante alto (aunque los últimos modelos aparecidos en el mercado presentan características de error bastante mejoradas). En consecuencia, el usuario sólo es libre de moverse dentro

de un cierto volumen de unos 2 o 3 metros alrededor del emisor. Para la última generación de dispositivos, más precisos, dichos errores son del orden del 10% a 2 m de distancia del emisor.

- En segundo lugar, el error se incrementa sobremanera en presencia de objetos metálicos de un cierto tamaño, o por la proximidad de equipos electrónicos, especialmente las pantallas de ordenador. Como resultado, el alcance efectivo del sistema disminuye, al incrementarse el error relativo de las medidas.
- El tercer problema es más grave aún, y es que las medidas realizadas presentan un cierto retardo, es decir, el dispositivo de localización tarda un cierto tiempo en recibir y procesar las señales utilizadas para determinar una cierta posición. Dicho retardo suele estar comprendido entre 20 y 80 mseg, aunque puede llegar a ser de hasta 150 mseg con algunos productos. Eso quiere decir que, desde que un usuario inicia un cierto movimiento hasta que el programa de simulación es notificado, pueden pasar entre 20 y 150 mseg. El resultado es que el usuario ve que su propio movimiento y el de la imagen no están sincronizados, lo que deteriora bastante la ilusión de inmersión y puede provocar la aparición del mareo de simulación.

Sin embargo, este tipo de dispositivos tiene también diversas ventajas, que hacen que sean los más utilizados en la actualidad: permiten libertad de movimiento total al usuario dentro del rango de actuación del sistema, tienen un coste relativamente bajo, no necesitan que haya una línea directa de visión entre los receptores y el emisor, pueden acoplarse fácilmente a cualquier dispositivo de visualización y, sobre todo, no son meros prototipos, sino que están ya disponibles comercialmente.

4.1.2.1. Sistemas comerciales

Los más conocidos entre los dispositivos electromagnéticos de localización comercializados son los de la serie 3SPACE de la empresa Polhemus y los de la firma Ascension Technologies, cuyo nombre comercial es A Flock of Birds (Una Bandada de Pájaros).

La serie 3SPACE de Polhemus la componen los siguientes productos:

- Isotrak II: consta de un emisor y uno o dos receptores. Su rango de acción es de aproximadamente 1.5 m, su frecuencia es de 60 Hz (30Hz si se usan los dos receptores), y el retardo de unos 50 mseg. Su precio, para un único receptor, es de unos 2900$.
- Fastrak: consta de 1 emisor y hasta 4 receptores. El rango de acción es de unos 3m. La frecuencia es de 120Hz dividida por el número de receptores utilizados (es decir, si se usan 4 receptores, la frecuencia se reduce a 30 Hz), y el retardo es de unos 10 mseg. Su precio, para un único receptor, es de 5800$.
- Insidetrak: el último equipo de la gama. Es una tarjeta que se inserta directamente en cualquier ranura ISA de un PC. Con un emisor y uno o dos receptores, su frecuencia de muestreo y su rango son iguales a los del Isotrak II, pero su retardo es la mitad de aquél, unos 25 mseg. Su precio, para un receptor, está en 2300$.

Polhemus fabrica, asimismo, diversos dispositivos basados en su tecnología de localización, como tabletas y lápices digitalizadores, o ratones tridimensionales.

Los dispositivos de localización de Polhemus son los mas extendidos, ya que fue la primera compañía en sacar al mercado este tipo de producto. Como resultado, muchos otros productos comerciales de realidad virtual emplean los sensores de Polhemus como mecanismo de localización, por ejemplo los guantes Dataglove o los visiocascos EyePhone de VPL, y la mayoría de los paquetes software de desarrollo de aplicaciones de realidad virtual tienen drivers para los sensores de Polhemus.

La firma Ascension Technologies fabrica y comercializa A Flock of Birds, un sistema de posicionamiento electromagnético que es la competencia directa de los de Polhemus. A Flock of Birds presenta la desventaja de que cada unidad de control sólo puede controlar un receptor (aunque pueden fácilmente conectarse entre sí hasta 30 unidades de control, lo que proporciona 30 receptores). Su alcance es similar al Fastrak de Polhemus. Presenta una frecuencia de muestreo superior (144 Hz), así como menor susceptibilidad a la presencia de objetos metálicos.

4.1.3. Dispositivos mecánicos

Los dispositivos mecánicos utilizan un brazo con múltiples articulaciones para determinar la orientación y posición del usuario. Uno de los extremos del brazo articulado se encuentra fijo a un cierto punto, mientras que el otro está adherido al dispositivo de visualización empleado o a la cabeza del usuario. La figura 3.2 muestra uno de tales dispositivos, empleado en un sistema binocular.

Puesto que la longitud de los elementos del brazo articulado es fija, basta con medir los ángulos de cada una de las articulaciones para conocer con exactitud la posición y orientación del extremo. Para medir los ángulos se utilizan potenciómetros o sensores ópticos.

Cuando el usuario quiere variar su punto de vista dentro de la escena, o desplazarse, tiene que mover físicamente el dispositivo de visualización, lo que hace que las articulaciones del brazo mecánico giren. Una unidad de control se encarga de realizar los cálculos, transmitiéndose los datos al ordenador, generalmente a través del puerto serie.

Este sistema tiene la ventaja de su gran precisión y carencia de retardos (la latencia es del orden de un par de milisegundos), pero se trata de un sistema en el que los movimientos del usuario están más limitados: no resulta posible realizar movimientos relativamente bruscos, debido al juego de las articulaciones, y el alcance está limitado por la longitud de las mismas.

Como consecuencia, este tipo de dispositivo no ha tenido una excesiva aceptación comercial hasta el momento. El BOOM de Fake Space Labs, del que ya hemos hablado en el capítulo dedicado a los dispositivos de salida, es el más conocido de los sistemas que emplean dispositivos mecánicos de localización. En el BOOM, el brazo articulado forma un todo con el dispositivo de visualización, que el usuario maneja como si fuera un periscopio.

La empresa canadiense Shooting Star Technology, por su parte, vende el sistema ADL-1 de bajo coste (en torno a los 1.500$), que es un brazo articulado acabado en una especie de

banda que el usuario se coloca alrededor de la frente. Su aplicación principal, en consecuencia, está en el campo de los sistemas de sobremesa.

4.1.4. Dispositivos ultrasónicos

En los dispositivos de localización ultrasónicos, se emplea un procedimiento de triangulación puro. Un emisor produce un ultrasonido que es recogido en tres receptores dispuestos triangularmente. Cuando el emisor produce la señal, se arranca un contador en los tres receptores, de forma que cada uno de éstos puede medir el tiempo que la señal tarda en llegar hasta él. A partir de ahí y del valor conocido de la velocidad del sonido, se determina la posición del emisor con respecto a cada uno de los receptores y, por triangulación, se obtiene la posición absoluta del emisor.

Esta técnica de localización presenta varias desventajas. En primer lugar, se necesita una línea directa de visión entre el emisor y los receptores, por lo que cualquier cuerpo (como el brazo del usuario) puede impedir la medición. En segundo lugar, el emisor y los receptores son direccionales, por lo que esta técnica no es válida más que en los casos donde el usuario esté siempre mirando (aproximadamente) hacia el lugar donde los receptores estén situados. En tercer lugar, permite únicamente medir posiciones, pero no orientaciones. Finalmente, esta técnica está sujeta a los mismos problemas de latencia que la localización por medio de dispositivos electromagnéticos.

A cambio, los dispositivos ultrasónicos son baratos, no se ven interferidos por la presencia de objetos metálicos, y sirven perfectamente en los casos en que no se necesita medir orientaciones y el usuario está siempre mirando en la misma dirección, como es el caso de las aplicaciones de sobremesa. Por eso se basan en esta técnica distintos dispositivos, como los ratones 3D actualmente existentes en el mercado o el dispositivo de localización para aplicaciones de sobremesa HeadTracker, de Logitech.

4.1.5. Otros sistemas de localización

Actualmente hay algunas otras tecnologías en fase de investigación que aún no han producido ningún resultado comercial. Existen sistemas ópticos que utilizan técnicas de procesamiento digital de imágenes para detectar la posición de determinadas partes del cuerpo: la imagen del usuario es capturada por una o más cámaras y analizada mediante un ordenador. De esta forma, puede determinarse la posición del usuario y de las distintas partes de su cuerpo. En la actualidad, esta técnica se ha aplicado con éxito en el caso de dos dimensiones (es decir, capturando una sola imagen bidimensional del usuario y analizando sus movimientos en el plano), existiendo algunos sistemas experimentales donde el usuario comunica de esta forma los comandos de control al sistema. Pero todavía dista mucho esta técnica de poder utilizarse como sistema de localización.

Algunos otros sistemas más elaborados emplean diodos electroluminiscentes o marcas de determinados colores en distintas partes del cuerpo del usuario, lo que facilita la tarea de análisis de la imagen. Esta técnica se emplea, por ejemplo, en sistemas donde se hace gesticular a una cara artificial siguiendo los gestos de un actor real.

Otros sistemas actualmente en fase de investigación son los inerciales, que utilizan giroscopios para medir la orientación de los objetos y acelerómetros para conocer su aceleración. Conocida la aceleración, puede obtenerse por integración la velocidad y, a partir de ésta, la posición del objeto, aunque los errores se van acumulando a lo largo del tiempo. Entre los primeros equipos comerciales basados en esta tecnología que han salido al mercado se encuentra el GyroPoint de la empresa californiana Gyration.

También se está investigando con la posibilidad de emplear compases electrónicos (que miden la dirección del campo magnético terrestre) para tareas de localización, habiendo ya en el mercado empresas (como Precision Navegation) que ofrecen módulos de bajo coste con esta funcionalidad, para su integración en sistemas de RV.

4.2. DISPOSITIVOS DE CONTROL

Los dispositivos de control son los que permiten al usuario realizar el control explícito de las aplicaciones e interactuar con el mundo virtual.

Dentro de un sistema de RV, el tipo de órdenes transmitidas varía enormemente de una aplicación a otra, pero se pueden distinguir tres grandes categorías:

- **Comandos de navegación**, mediante los que el usuario indica al sistema que desea desplazarse dentro del mundo virtual.
- **Comandos de interacción**, que permiten al usuario interactuar con los elementos que componen el mundo virtual. El usuario puede desear, por ejemplo, agarrar un objeto, desplazarlo, o efectuar determinadas acciones como disparar o tocar un instrumento virtual.
- **Comandos de manipulación del estado** del mundo virtual, mediante los que el usuario puede regular determinadas variables de estado que afectan al funcionamiento global del sistema. Normalmente, este tipo de comandos se implementará mediante un menú o cualquier otro sistema por el que el usuario pueda elegir entre una serie de opciones, o introducir un valor.

El tipo de dispositivo empleado para cada categoría de comandos es muy variable, y dependerá de la aplicación concreta que se esté ejecutando y de la configuración que para el sistema se haya elegido. A menudo, un mismo dispositivo se suele usar para transmitir más de un tipo de comando. Por ejemplo, un electroguante puede emplearse para ordenar un desplazamiento, apuntando con el dedo en una determinada dirección, y también para agarrar un determinado objeto de la escena.

En determinadas aplicaciones, el tipo de dispositivo que se empleará viene determinado por la propia naturaleza de la aplicación. Así, en los equipos de simulación (por ejemplo un simulador de conducción) lo más apropiado es emplear dispositivos similares a los que se encuentran en el caso real: volante, acelerador, freno, ... con el objeto de aumentar el realismo de la aplicación y entrenar lo más perfectamente posible al usuario en el manejo del equipo real.

En este apartado, vamos a pasar revista a los principales dispositivos de control utilizados en las aplicaciones de RV. Vamos a centrarnos en aquéllos que son más característicos de los sistemas de RV, dejando a un lado los dispositivos más convencionales.

Los dispositivos de control convencionales, como joysticks, ratones y teclados, se emplean profusamente en multitud de aplicaciones de RV como dispositivos de navegación y control. De hecho, hay multitud de aplicaciones en las que los dispositivos especializados no son imprescindibles, de forma que muchos sistemas de bajo coste emplean sólo controles convencionales por razones económicas.

Hay además algunas circunstancias en que el uso de dispositivos convencionales es prácticamente obligado. Es el caso, por ejemplo, de las aplicaciones donde es preciso proporcionar información textual al sistema, en cuyo caso el teclado sigue siendo (hoy por hoy) la única alternativa viable. O también el caso de determinados sistemas de sobremesa donde la ventana al mundo 3-D no ocupa toda la pantalla, sino que está enmarcada por una zona 2-D reservada al control de la aplicación (por ejemplo, una serie de menús o iconos de control). En este caso, un ratón convencional permite efectuar cómodamente las tareas de control, pudiendo emplearse dispositivos más sofisticados para la interacción dentro de la ventana 3-D.

Sin embargo, el lector conoce de sobra el funcionamiento y aplicaciones de este tipo de dispositivos, por lo que nos centraremos en aquéllos otros más específicos de los sistemas de RV.

4.2.1. Electroguantes

Los electroguantes son dispositivos utilizados en las aplicaciones de RV para detectar la posición de la mano del usuario. Lo que los electroguantes miden es, básicamente, la posición relativa de los dedos, detectando el grado de flexión de cada uno y la separación existente entre dedos consecutivos.

Los electroguantes se suelen realizar en nylon o lycra, e incorporan una serie de dispositivos sensores que permiten determinar el grado de flexión y la separación (abducción) entre los dedos. Los datos suministrados por el electroguante se recogen en el ordenador, generalmente a través del puerto serie, y se procesan para extraer la información sobre las posiciones relativas de cada articulación. Incorporando un dispositivo de localización al electroguante, se puede complementar dicha información con la relativa a la posición absoluta de la mano.

4.2.1.1. Aplicaciones

Los electroguantes pueden emplearse en tres diferentes situaciones:

a) La aplicación más intuitiva consiste en utilizar los datos suministrados por el guante para simular en pantalla una mano virtual. La mano virtual sigue exactamente, gracias a los sensores, los movimientos de la mano del usuario. Eso permite agarrar

Figura 4.2. Electroguante.
(Fotografía de Alejandro Sacristán)

objetos en el mundo virtual, tocar un instrumento virtual o, en general, realizar cualquier tipo de acción en la que se requiera la intervención de la mano.

b) El guante también puede emplearse para controlar objetos complejos, como un instrumento o el brazo articulado de un robot virtual, sin necesidad de que se visualice una mano virtual en la pantalla. En este caso, se establece una correspondencia (a nivel software) entre los movimientos de los dedos y determinadas operaciones de los objetos virtuales. Como ejemplo, podemos definir una correspondencia entre determinadas acciones y los gestos de la mano, de forma que el sistema emplee los datos de posición de los dedos para regular el funcionamiento de un brazo mecánico.

La no existencia de la realimentación visual directa implica que el usuario necesita aprender el código de control del objeto (es decir, qué movimiento de la mano corresponde a cada acción), por lo que este tipo de uso está reservado a aplicaciones de carácter profesional o semiprofesional, y especialmente al campo de la tele-robótica y el telecontrol.

Cabe destacar, con respecto a este tipo de aplicación, las investigaciones llevadas cabo en el Trinity College de Dublin sobre lenguajes gestuales para electroguantes, dentro del proyecto GLAD-IN-ART, en el marco del programa europeo ESPRIT.

c) Finalmente, el guante puede emplearse como un instrumento de control de propósito general, igual que un joystick o un ratón. Por ejemplo, el usuario puede apuntar con

un dedo para indicar en qué dirección desea desplazarse, para seleccionar un objeto del mundo virtual o para elegir una opción de un menú. En ese caso, es responsabilidad del software determinar qué acciones son posibles en cada momento, qué movimientos de la mano corresponden a cada una, y qué tipo de información visual (de ayuda y de realimentación) recibirá el usuario.

Normalmente, esta aplicación de los electroguantes nunca se presenta sola, sino en conjunción con una de las dos aplicaciones anteriores. El motivo es que, si el electroguante sólo va a usarse para este propósito, es mucho más económico emplear un dispositivo convencional, como un ratón o un joystick.

Los electroguantes tienen algunas otras aplicaciones no estrictamente relacionadas con la realidad virtual. En el centro de investigación Loma Linda, en EEUU, emplean el electroguante como un puro instrumento para medir la intensidad de los temblores en pacientes aquejados de Parkinson. Asimismo, Virtual Technologies, fabricante del electroguante CyberGlove, está desarrollando una aplicación en la que, utilizando una red neuronal, se reconocen los gestos efectuados con la mano. Con ella, una persona muda puede controlar un sintetizador de voz utilizando un lenguaje de gestos.

4.2.1.2. Tecnologías existentes

Existen dos formas fundamentales de medir la flexión y abducción de los dedos. La primera se basa en la utilización de sensores flexibles. El sistema utilizado por los guantes Dataglove, de la empresa VPL, consiste en unas fibras ópticas que se incorporan al guante, de tal manera que las fibras se flexionan cuando el usuario abre o cierra los dedos. La luz que pasa a través de las fibras ópticas se atenúa al doblarse la fibra, con lo que puede determinarse el grado de flexión a partir de la cantidad de luz que consigue atravesar. Utilizando los correspondientes dispositivos optoelectrónicos, dicha información se traduce a una señal eléctrica que el ordenador es capaz de procesar.

La segunda forma consiste en usar una estructura de segmentos articulados, que se fija a la mano como una especie de exoesqueleto. La flexión de los dedos origina una variación en el ángulo que los segmentos forman entre sí. Mediante un potenciómetro, o mediante dispositivos ópticos, dichos ángulos pueden medirse con bastante precisión, lo que permite determinar el grado de flexión o abducción de los dedos. Este es el método empleado en el Dexterous Handmaster, de la empresa Exos.

Los guantes que utilizan sensores flexibles son más cómodos de emplear, puesto que pueden ponerse muy sencillamente, mientras que los que utilizan exoesqueleto son bastante más engorrosos. A cambio, estos últimos tienen un mayor grado de precisión.

Tanto unos como otros requieren una recalibración del guante para cada persona concreta. En los guantes que emplean sensores flexibles, es necesario recalibrar por separado cada sensor, debido a las diferencias de materiales.

Las características que determinan las diferencias entre unos guantes y otros son, fundamentalmente, el número de parámetros distintos que pueden medir (cuántas medidas de las articulaciones, y si pueden medir también la abducción), y la frecuencia de muestreo.

4.2.1.3. Sistemas comerciales

Entre los guantes comerciales más extendidos podemos citar:

- El Cyberglove, de Virtual Technologies. Utiliza galgas extensiométricas para medir 22 parámetros distintos de flexión y abducción de los dedos, con una frecuencia de muestreo máxima de 100 Hz.
- El DHM, de Exos, que emplea semiconductores de Efecto Hall como dispositivo sensor, permitiendo medir 20 parámetros diferentes.
- El DataGlove, de VPL, en el que se emplean fibras ópticas para medir 10 parámetros distintos de flexión de los dedos, con una frecuencia de muestreo de 160 Hz. El DataGlove incorpora un dispositivo de localización de Polhemus para poder medir la posición y orientación de la mano. La compañía juguetera Mattel comercializó durante algún tiempo un guante de bajo coste derivado del DataGlove para su utilización con las consolas Nintendo, aunque ya ha dejado de fabricarlo.

El concepto de los electroguantes como dispositivos de control es fácilmente extensible, pudiéndose emplear la misma tecnología para medir las flexiones de otras partes del cuerpo, como el codo o la muñeca. La empresa Virtual Technologies fabrica y comercializa distintos dispositivos basados en esta técnica, incluyendo el CyberSuit, un traje completo para detectar la postura en la que el usuario se encuentra. La empresa VPL fabrica otro dispositivo similar, el Datasuit.

4.2.2. Ratones y joysticks 3-D

Ya hemos mencionado que los sistemas de RV emplean profusamente dispositivos convencionales para tareas de navegación y control. El problema fundamental de los dispositivos convencionales, sin embargo, es que están mal adaptados a los mundos de tres dimensiones, lo que en ciertas aplicaciones puede obligar a emplear otros mecanismos de control.

Precisamente por esta inadecuación de los dispositivos de control convencionales a los mundos 3-D fue por lo que se desarrollaron una serie de dispositivos pensados específicamente para este tipo de aplicación.

Los ratones 3-D son una extensión natural de los ratones convencionales. Un ratón convencional es un dispositivo capaz de moverse sobre una superficie plana, de forma que puede utilizarse para controlar el movimiento de cualquier objeto en dos dimensiones, como es el caso de un cursor sobre una pantalla. De forma similar, un ratón 3D permite controlar el movimiento de cualquier objeto en un espacio tridimensional.

Lo único que hace falta para construir un ratón 3-D es un dispositivo de localización que permita al sistema determinar la posición del ratón, y una serie de pulsadores. El usuario lleva en la mano el dispositivo y lo mueve en tres dimensiones a voluntad.

Puesto que lo único que interesa es la posición del ratón, los dispositivos de localización ultrasónicos son los más adecuados, por su bajo coste. Ésta es la solución adoptada en el 3D

Mouse, de la empresa Logitech, que es el más popular de estos dispositivos. Pero los electromagnéticos o mecánicos son igualmente válidos, desde el punto de vista técnico.

El 3D Mouse de Logitech presenta una resolución de 200 puntos por pulgada, tiene una tasa de actualización de 50 Hz, y se conecta al ordenador a través del puerto serie. Otras empresas que comercializan ratones diseñados para aplicaciones 3D son Mouse Systems y Polhemus.

En la actualidad se está trabajando en el desarrollo de dispositivos mejor adaptados a las aplicaciones en tres dimensiones. Así, además del ratón 3-D, existen joysticks tridimensionales, como el de la empresa francesa Virtools, o dispositivos como el SpaceBall, de la empresa americana SpaceBall Technologies, que combina la funcionalidad de un ratón 3D con la posibilidad de medir la fuerza que el usuario aplica. De esta manera puede controlarse con un sólo mando no sólo la dirección de movimiento, sino también la velocidad.

4.2.3. Dispositivos para simuladores

Existe otra categoría de dispositivos de entrada que no por ser poco sofisticados carecen de importancia. Se trata de los dispositivos mecánicos empleados en las aplicaciones de simulación.

En los simuladores de vuelo, simuladores de conducción, simuladores militares de vehículos acorazados, etc... es común utilizar cabinas similares a las reales, con el fin de entrenar a los usuarios en condiciones lo más parecidas posible a las de funcionamiento normal. En dichas cabinas, el usuario dispone de los mismos (o parecidos) controles mecánicos que en la realidad, y es a través de ellos como transmite sus órdenes al sistema.

Pero no sólo sucede esto en aplicaciones profesionales, sino que tenemos infinidad de ejemplos de este tipo de sistema en las galerías de videojuegos. Mientras que en los simuladores profesionales se emplean controles reales porque de lo que se trata precisamente es de entrenar al usuario en su manejo, en las cabinas a medida de ciertos videojuegos se emplean controles casi reales con el fin de acrecentar el realismo (y, por consiguiente, el interés) del juego.

El integrar dichos dispositivos mecánicos en un sistema de realidad virtual no presenta mayor problema desde el punto de vista técnico que incorporar los apropiados sensores. Para la aplicación, todos los dispositivos de este tipo se pueden modelar, en última instancia, como si fueran botones o joysticks.

4.2.4. Sistemas bioeléctricos

Saliendo del campo experimental del laboratorio para adentrarse de lleno en el terreno comercial, los sistemas bioeléctricos comienzan a perfilar lo que constituye una de las líneas de futuro más prometedoras dentro del mundo de la realidad virtual.

Hace muchos años que se procesan señales bioeléctricas con el fin de monitorizar ciertas constantes vitales, como en el caso de los electrocardiogramas. Pero sólo recientemente se ha materializado la idea de que dichas señales pueden usarse para controlar un dispositivo.

Los sistemas bioeléctricos miden distintas señales eléctricas generadas por el cuerpo humano, y las transforman en señales de control analógicas y digitales, que pueden emplearse para gobernar cualquier tipo de dispositivo.

La empresa BioControl Systems, de California, es la pionera en el desarrollo de este tipo de dispositivo. Quizá el más llamativo de los fabricados por dicha compañía sea el controlador ocular Eyecon. Este dispositivo detecta y procesa las señales generadas por los movimientos del ojo (electro-óculo-grama), y las transforma en señales de control que pueden utilizarse en cualquier aplicación. Con dicho dispositivo se puede, por ejemplo, gobernar el funcionamiento del ratón en la pantalla con el simple movimiento de los ojos.

La principal aplicación de este tipo de dispositivo es en el control del nivel de detalle de las imágenes sintetizadas por un sistema de RV, como veremos en el próximo capítulo.

También las señales generadas por los músculos (electro-miograma) pueden ser utilizadas de esta forma. Los equipos Enabler y Biomuse RF, de la misma compañía, además de utilizar el electro-óculograma como elemento de control, incorporan sensores que pueden emplearse para detectar las señales de músculos diferentes del cuerpo, las cuales son procesadas y transformadas en señales de control. Las aplicaciones de este tipo de dispositivo son múltiples, desde la ayuda a personas impedidas (por ejemplo para manejo de una prótesis), hasta el control de brazos mecánicos.

Pero no son éstas las únicas señales que pueden detectarse. Otro dispositivo fabricado por Biocontrol Systems, denominado Brainman, permite detectar y transformar en señales de control las ondas cerebrales. La empresa Meta Sound Engineering fabrica otro dispositivo similar, el Interactive Brainwave Visual Analyzer. El problema con este tipo de dispositivo es que, mientras que tenemos un control consciente sobre nuestros ojos y músculos, no sucede lo mismo con las ondas cerebrales.

En un futuro posiblemente no muy lejano se podrán usar dispositivos que permitan, una vez que el usuario piense en un determinado comando, traducir dicho pensamiento a la correspondiente señal de control. Pero por el momento, los algoritmos existentes de análisis de las ondas cerebrales no permiten interpretar dicho tipo de comandos subvocalizados. Nuestro conocimiento de las ondas cerebrales no es suficiente como para discernir qué partes de ellas están sujetas a control consciente, con lo que no se las puede usar como señales de control voluntarias. Por el momento, lo más que se ha conseguido es emplearlas como entrada para un sintetizador de música.

Sin embargo, la investigación en este sentido ya ha comenzado. La compañía The Artisan Group, en colaboración con el Centro Tecnológico Westinghouse, en Pittsburgh, está trabajando en el análisis de las ondas cerebrales, con el fin de identificar posibles patrones que puedan ser voluntariamente controlados por el individuo.

4.3. OTROS DISPOSITIVOS DE ENTRADA

Vamos a cerrar este capítulo con el repaso de otros tipos de dispositivos utilizados en aplicaciones de RV.

- Dispositivos de **adquisición de datos**: como ya se ha mencionado anteriormente, los sistemas de RV pueden emplear diversos dispositivos para efectuar mediciones de parámetros del mundo real. Además de los dispositivos de localización, que permiten medir la posición y orientación del usuario o de un objeto, un sistema de RV puede utilizar cualquier otro tipo de sensor analógico o digital. Este tipo de dispositivo se emplea, fundamentalmente, en aplicaciones de visualización científica y tele-robótica.
- Dispositivos de **reconocimiento de voz**, como los que incorporan muchas tarjetas de sonido, desde la Soundblaster, de Creative Labs, a la SoundMaster II, de Covox o la SIR Model 20 de Scott Instruments. Las técnicas de reconocimiento de voz tienen la ventaja sobre las técnicas tradicionales de que no exigen que el usuario emplee las manos, con lo que éstas quedan libres para otras tareas. Las desventajas son que se requiere un adiestramiento previo, para enseñar al dispositivo a reconocer las palabras al modo en que cada usuario concreto las pronuncia, y que representa un engorro si el usuario comparte el área de trabajo con otras personas. Por lo demás, las técnicas de reconocimiento de voz están bastante depuradas hace ya mucho tiempo, y están sujetas a errores mucho menores que, por ejemplo, el reconocimiento de gestos de las manos mediante electroguantes.
- Dispositivos de **reconocimiento de imágenes**: diversos organismos, como el Instituto de Investigación en Telecomunicaciones Avanzadas de Kioto (Japón), están desarrollando sistemas para el reconocimiento óptico de gestos y movimientos faciales. En la actualidad, existen ya múltiples aplicaciones en las que se reconocen los gestos utilizando unos adhesivos de colores que se sitúan en puntos determinados de la cara del sujeto. El reto está, en estos momentos, en conseguir un reconocimiento medianamente libre de errores sin utilizar ningún tipo de mecanismo adicional.
- **Dispositivos MIDI**: puesto que muchos sistemas de RV incorporan dispositivos de sonido capaces de sintetizar secuencias musicales a partir de comandos MIDI, puede emplearse un canal MIDI para el control en tiempo real de las secuencias sonoras. Como ejemplo, el programa More Bells & Whistles, desarrollado por Wayne Litle en el Cornell National Supercomputing Facility, es un programa donde se puede controlar mediante comandos MIDI una secuencia de animación que representa instrumentos musicales. Esta idea ya había sido ensayada por Robert Sabiston en el MIT en 1988. Aunque la animación es en 2-D, su generalización a 3-D es inmediata.
- **Controladores de fuerza/giro**: son dispositivos similares en funcionalidad a los joysticks, pero incorporan realimentación cinestésica. Cuando el usuario acciona el mando de un controlador de fuerza/giro en una dirección, la fuerza que debe aplicar es mayor cuanto más grande sea el desplazamiento. De forma similar, el usuario puede cambiar la orientación del mando, y la fuerza que deberá aplicar crecerá a medida que el ángulo se incremente. Este tipo de dispositivo encuentra su aplicación principal en simuladores, puesto que la realimentación cinestésica contribuye a un control más preciso del usuario en las tareas de navegación.

5
Síntesis de imágenes 3-D

Una vez efectuado un repaso a los dispositivos con los que un sistema de RV se comunica con el mundo exterior, hora es ya de que volvamos nuestra atención al interior del sistema, para tratar de comprender su funcionamiento.

En el capítulo siguiente, expondremos en detalle el funcionamiento de una estación de proceso típica de un sistema de RV. Pero entre los cometidos de la estación de proceso, destaca especialmente uno, la función de la síntesis de imágenes, cuya complejidad justifica por sí sola el que le dediquemos un capítulo específico.

A lo largo de los próximos apartados vamos a analizar el proceso de generación de las imágenes en una aplicación de RV, desde el momento en que el sistema decide cuál es el correcto punto de vista de la imagen, hasta el momento en que ésta queda plasmada en una serie de pixels sobre una o varias pantallas.

El objetivo del capítulo no es hacerle conocer los fundamentos matemáticos del proceso de síntesis de imágenes, sino explicar en qué consiste éste y cuáles son los conceptos implicados. Si alguna vez tiene que diseñar una aplicación de RV, usted no va a tener que preocuparse de escribir un programa para implementar la síntesis ni de realizar los complejos cálculos involucrados. Lo más probable es que su sistema incorpore alguna librería software que ya tendrá implementados los algoritmos necesarios. Pero aunque no se entre en el detalle matemático de cada paso del proceso de síntesis, lo que sí es necesario es conocer en qué consiste y cuáles son los principales factores que inciden en su rendimiento. El conocer los factores que determinan el rendimiento del proceso de la síntesis es imprescindible por dos motivos diferentes:

- En primer lugar, porque necesitamos tener elementos de juicio para determinar, a la hora de diseñar el sistema, qué elementos hardware y software se adaptan mejor a nuestras necesidades. El conocer en qué se basa el proceso de síntesis de imágenes nos ayudará, por ejemplo, a comparar una librería gráfica con otra, o dos estaciones de proceso, o dos tarjetas gráficas.
- En segundo lugar, porque el conocer los factores que influyen en el rendimiento del sistema nos permitirá sacar el máximo partido de la configuración hardware y software seleccionada. Conociendo los requerimientos de cada paso del proceso de síntesis podemos responder preguntas como: ¿puedo aumentar el número de objetos dentro

de una escena determinada, sin que el sistema se sature? ¿puedo incrementar el nivel de detalle de los objetos? ¿debo emplear alguna técnica especial para reducir el tiempo de generación de la imagen?

En este capítulo nos vamos a ocupar de analizar cuál es el proceso utilizado para sintetizar una imagen estática. Es decir, vamos a ver cómo los sistemas de RV generan en cada momento la imagen a partir de la definición estática de una escena. En el capítulo próximo veremos, al hablar de las funciones de la estación de proceso, lo referente a cómo implementar animaciones, es decir, sucesiones de imágenes.

5.1. EL PROCESO DE SÍNTESIS (I)

En primer lugar, vamos a ver qué entendemos por sintetizar una imagen. El proceso de síntesis de la imagen (denominado en la literatura inglesa "rendering") es la secuencia de pasos que, partiendo de la definición gráfica de los objetos y la posición de éstos dentro de la escena, permite generar la imagen que representa la visión del mundo virtual correspondiente a la posición que el usuario ocupa.

Hay que hacer notar que el proceso de síntesis comienza una vez que el proceso de simulación ha concluido: para poder mostrar al usuario una nueva imagen, es necesario que el sistema haya efectuado previamente todos los cálculos necesarios para determinar la posición de cada objeto (incluido el usuario) dentro de la escena. Una vez que esas posiciones han sido determinadas, la imagen final que se le presente va a depender de dos factores distintos: la geometría de la escena y el modelo de iluminación.

La **geometría de la escena** viene definida por la forma de los objetos y por las posiciones que éstos ocupan dentro del mundo virtual. Podríamos decir que ésta es la materia prima de la que parte la síntesis de imágenes. Son los datos de entrada al proceso de la síntesis.

Podemos considerar, simplificando un poco, que el proceso de síntesis toma como dato de entrada una lista de objetos (la geometría de la escena), generando como dato de salida la imagen que debe mostrarse en la pantalla. La lista de objetos indica qué posición ocupa cada objeto dentro de la escena y cuál es su representación gráfica. La imagen de pantalla, por su parte, no es más que otra lista que indica el color que debe darse a cada uno de los pixels que componen la pantalla.

Pero la imagen final que se presente al usuario no va a depender sólo de esos datos geométricos de entrada. Dependiendo de cómo elijamos "iluminar" la escena, (es decir, de nuestro **modelo de iluminación**) la imagen final puede variar enormemente. Podemos, por ejemplo, emplear una única fuente luminosa o introducir un conjunto de ellas. Podemos hacer que los objetos "proyecten" sus sombras sobre el suelo u otros objetos o, por el contrario, prescindir de los complejos cálculos que estas sombras requieren. Podemos hacer que los objetos reflejen la luz de una manera u otra; podemos regular el nivel de detalle de los cálculos de iluminación, ... Todos estos factores van a incidir en la calidad final de la imagen tanto o más que la propia geometría de los objetos. Aun cuando empleáramos un gran nivel de detalle a la hora de definir la forma de nuestros objetos, un modo de iluminación pobre puede dar al traste con cualquier sensación de realismo.

Básicamente, el proceso de síntesis consiste en proyectar los objetos sobre la pantalla, y calcular el color que cada punto de la superficie del objeto debe tener. Mientras que la proyección del objeto en pantalla va a depender de la forma y posición del objeto (es decir, de la geometría de la escena), el color que demos a cada punto dependerá de cuál sea el color del objeto, pero también de dónde estén situadas las fuentes de luz, del tipo de éstas y, en definitiva, del modelo de iluminación utilizado. Un objeto de color uniforme (rojo, por ejemplo) no se presenta de manera uniforme en la imagen final: las zonas situadas de cara a una fuente de luz, por ejemplo, aparecerán más brillantes que las zonas donde la luz no incide.

En los siguientes apartados describiremos el proceso de síntesis, primero en una versión simplificada y, por así decirlo, estándar, para luego añadir poco a poco complicaciones adicionales. Pero antes de empezar conviene que echemos un vistazo al punto de partida y a los pasos previos que el proceso de síntesis de las imágenes requiere.

5.2. GEOMETRÍA DE LAS ESCENAS

5.2.1. Definición geométrica de los objetos

De cara a la representación de escenas, todos los objetos visualizables tienen asociada, en cada momento, una definición gráfica concreta.

Esa definición gráfica indica, como mínimo, cuál es la forma del objeto, aunque lo normal es que incluya otros datos, dependiendo de los requerimientos de la aplicación: información de color, texturas, coeficientes de reflexión, etc...

La forma de los objetos se expresa mediante una lista de los polígonos que los componen, donde cada polígono queda definido por las coordenadas de sus vértices. Como ejemplo, la figura 5.1 muestra un cubo cuya longitud de arista es la unidad. La definición gráfica del cubo contendría los datos que definieran cada una de sus seis caras. La definición para las tres primeras caras del cubo sería:

Polígono 1: (0,0,0) (0,1,0) (1,1,0) (1,0,0)

Polígono 2: (1,0,0) (1,1,0) (1,1,1) (1,0,1)

Polígono 3: (1,1,0) (0,1,0) (0,1,1) (1,1,1)

Obsérvese que las coordenadas de los vértices se definen con respecto a un origen de coordenadas arbitrario. Podríamos perfectamente haber situado dicho origen en un punto diferente, sin que eso cambie para nada la forma del objeto. Ese origen de coordenadas, sea cual sea, lo denominaremos **centro del objeto** (aunque esté situado fuera del objeto o, como en este caso, en un borde del mismo). El sistema de coordenadas correspondiente, con respecto al cual se definen los vértices, recibe el nombre de **sistema de coordenadas del objeto**.

Aunque en el ejemplo hemos utilizado cuadrados, normalmente suelen descomponerse los objetos en triángulos, porque eso facilita las últimas etapas del proceso de síntesis.

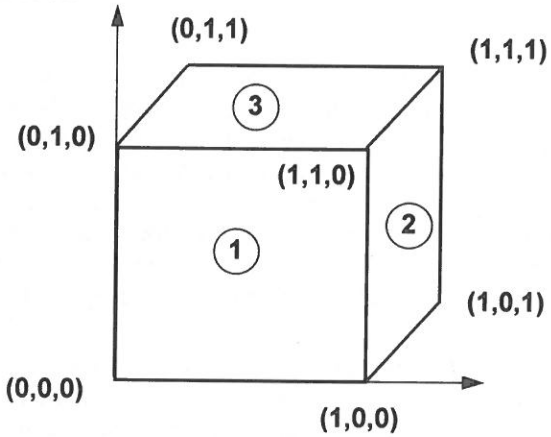

Figura 5.1. Definición geométrica de un cubo

Para objetos que no sean perfectamente poliédricos, y que tengan por tanto alguna superficie curva, lo que se hace es efectuar una aproximación a la superficie mediante un conjunto de polígonos (ver figura 5.2), y definir dichos polígonos mediante sus vértices, como en el caso anterior. La aproximación será tanto mejor cuanto mayor sea el número de polígonos que empleemos. Más adelante, en este mismo capítulo, veremos los efectos que tiene el efectuar tal aproximación.

5.2.2. Definición geométrica de las escenas

La definición geométrica de una escena será, básicamente, una lista de objetos, con sus correspondientes posiciones:

Objeto 1	Posición 1
Objeto 2	Posición 2
...	...
Objeto n	Posición n

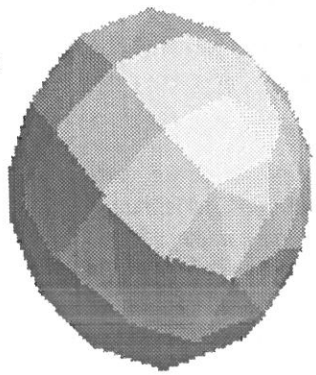

Figura 5.2. Aproximación poligonal a una esfera

Las posiciones de los objetos se expresan dando la posición del *centro del objeto*, y estarán definidas con respecto a un origen de coordenadas de la escena, al que denominaremos **centro de la escena**. El centro de la escena puede estar situado en cualquier punto arbitrario de la misma. El correspondiente sistema de coordenadas recibe el nombre de **sistema de coordenadas de la escena**.

Cuando decimos que hay que dar la *posición* del centro del objeto, no nos referimos sólo a sus coordenadas, porque eso sólo no basta: además, hay que definir cuál es la orientación del objeto (ver figura 5.3).

¿Ha oído usted hablar de las matrices? No, no se asuste, no vamos a entrar en complejas disquisiciones matemáticas. Sólo diremos que la forma de expresar ese cambio de orientación es, precisamente, mediante una matriz.

Conociendo esa matriz y las coordenadas del centro del objeto, queda perfectamente definida la posición de éste en el sistema de coordenadas de la escena.

Obsérvese que, hasta el momento, ya hemos hablado de dos sistemas de coordenadas distintas: los vértices que expresan la forma de un objeto están definidos en el **sistema de coordenadas del objeto**, y el objeto (como un todo) tendrá asignada su correspondiente posición en el **sistema de coordenadas de la escena**.

Más adelante, cuando hablemos de los pasos del proceso de síntesis, veremos que existen otros dos importantes sistemas de coordenadas: el **sistema de coordenadas del observador,**

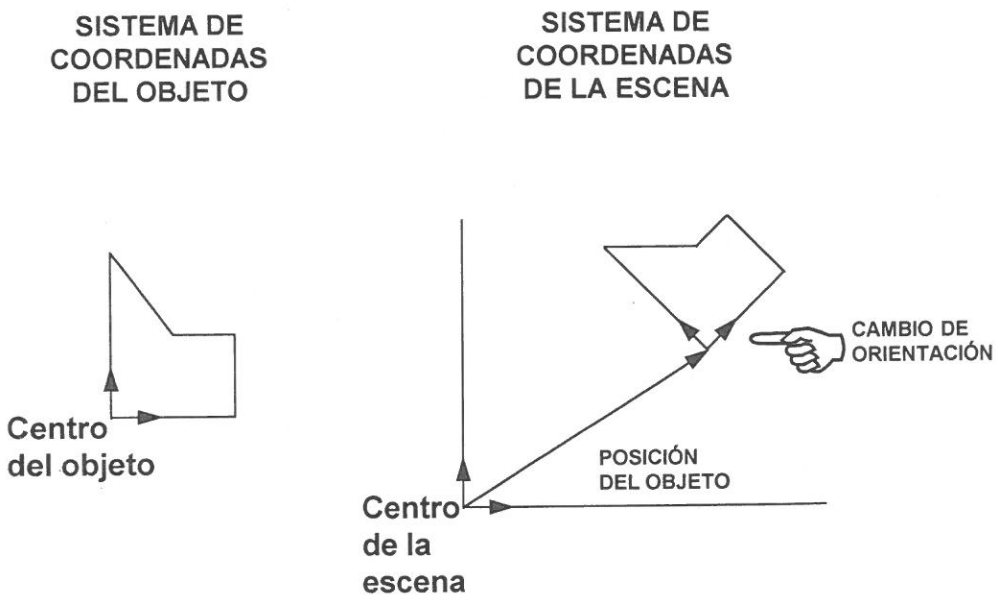

Figura 5.3. El sistema de coordenadas de la escena

y **el sistema de coordenadas de la pantalla.** También veremos que toda la primera parte del proceso de síntesis consiste, en esencia, en efectuar una serie de transformaciones entre estos cuatro sistemas de coordenadas.

5.3. MODELOS DE ILUMINACIÓN

En este apartado, vamos a exponer brevemente los aspectos fundamentales de los modelos de iluminación más utilizados en las aplicaciones de RV.

A la hora de mostrar un objeto en la pantalla en una aplicación de RV, es necesario calcular el color que cada punto del objeto tiene, de acuerdo con las fuentes de luz existentes en la escena, y de acuerdo con las características del propio objeto. El cómo calculemos dicho color va a venir determinado por el **modelo de iluminación** que elijamos.

El objetivo de una aplicación de RV no es conseguir escenas de una calidad fotográfica o de un realismo extraordinario, sino proporcionar un adecuado soporte gráfico a lo que es,

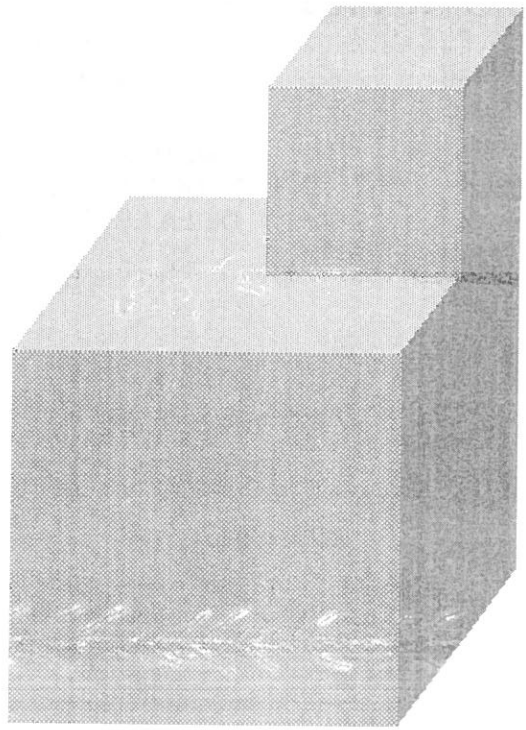

Figura 5.4. Sombreado en un objeto poliédrico

antes que nada, una simulación interactiva. El modelo de iluminación que se emplea, por tanto, suele ser bastante simple. Las restricciones que impone el procesamiento en tiempo real implican que no puedan emplearse modelos de iluminación complejos, como los que se utilizan en las animaciones comerciales de cine o TV. En los programas de RV, los gráficos son el medio; en una animación comercial, el fin.

El modelo de iluminación viene definido por varios factores: el tipo y número de las fuentes de luz, las propiedades de los materiales y el algoritmo de cálculo del sombreado.

Decimos "sombreado" y no "sombra". **Sombra** es la zona oscura que un objeto puede crear sobre el suelo o sobre otros objetos, al tapar una fuente de luz. El cálculo de las sombras puede ser bastante engorroso, por lo que la mayoría de las aplicaciones de tiempo real no contemplan este tipo de efecto. **Sombreado** es la diferencia de color que se produce entre unas zonas y otras de la superficie de un objeto, al ser éste iluminado. Las zonas del objeto donde la luz incide de frente aparecen con una intensidad de luz mayor, mientras que aquéllas donde la luz incide de manera oblicua adoptan un matiz más oscuro, y las zonas situadas en la parte de atrás del objeto (visto desde la fuente luminosa) resultarán completamente oscurecidas.

Al revés de lo que ocurre con las sombras, el sombreado sí es esencial en cualquier aplicación de RV. El motivo es que el sombreado es, precisamente, lo que nos permite percibir la forma de los objetos. Sin él, no podemos detectar las diferencias de inclinación de las distintas partes de la superficie de un objeto. La figura 5.4 es un ejemplo de cómo la perspectiva y el sombreado bastan para percibir la forma de los objetos. En la figura no se ha utilizado ni una sola línea; tan sólo se han sombreado con diferente intensidad las distintas caras del objeto.

5.3.1. Sombreado aleatorio

El modelo de iluminación más simple posible es el de **sombreado aleatorio**, utilizado en algunos videojuegos que emplean tecnología de gráficos 3D (como, por ejemplo, Virtua Fighter de Sega). Con el modelo de sombreado aleatorio, se selecciona de una forma esencialmente arbitraria un cierto nivel de luz para cada polígono de un objeto, y se ajusta el color de ese polígono de acuerdo con dicho nivel de luz. Así, un polígono de color rojo será de un rojo más vivo o más apagado dependiendo del nivel de iluminación que se determine para él.

El algoritmo de cálculo del color de un objeto con este modelo sería:

1) Dar a cada polígono un valor de iluminación arbitrario.
2) Determinar el color resultante del polígono según dicho nivel de iluminación.
3) Dar a todos los puntos del polígono el mismo color.

Este modelo tan simple es más que suficiente para permitir al cerebro humano distinguir las formas de los objetos. Sólo con que los polígonos adyacentes tengan una diferencia de

iluminación, el cerebro ya puede percibir los bordes de los polígonos y, por tanto, la estructura de un objeto.

En este modelo, no es necesario definir fuentes de luz ni ningún otro parámetro especial, y los cálculos que requiere son mínimos. Lo único que hay que cuidar es que el efecto estético sea el adecuado, lo que implica seleccionar los valores de iluminación de cada polígono con un cierto cuidado. Una vez seleccionado el valor de iluminación, todos los puntos de un determinado polígono tendrán el mismo color[1].

Aunque los resultados obtenidos con el sombreado aleatorio son más que suficientes para poder distinguir la forma de los objetos, el efecto final no es muy realista. En el mundo real, las zonas de sombra de un objeto están siempre situadas en la dirección contraria a la luz. Si el objeto cambia de orientación, las zonas que antes eran oscuras pueden pasar a estar iluminadas, y viceversa. Con un modelo de sombreado aleatorio, sin embargo, las zonas oscuras pueden estar situadas en cualquier lugar, y no guardan relación aparente con el movimiento del objeto.

5.3.2. Sombreado plano

En el modelo de sombreado plano, también se calcula un cierto nivel de iluminación para cada polígono y se da el mismo color a todos los puntos componentes del mismo. Pero, en lugar de calcularse arbitrariamente, el nivel de iluminación de cada polígono se determina de acuerdo a una serie de fuentes de luz predeterminadas.

Se pueden distinguir dos **tipos de fuentes de luz** fundamentales:

- **Luz ambiente**, que se puede considerar constante a lo largo de una escena. En el mundo real, las moléculas del aire desvían parte de la luz en todas las direcciones, por lo que incluso aquellas partes de los objetos sobre las que no incide directamente ningún rayo de luz están algo iluminadas.
- **Luz dirigida**, que es la procedente de un foco de luz concreto. Los rayos de luz procedentes de un foco concreto iluminarán sólo a aquellas superficies sobre las que incidan, y el grado en que las iluminen va a depender de la inclinación que las superficies tengan con respecto a la luz. Así, una superficie situada de cara a la luz (como está la Tierra con respecto al Sol en verano) se verá más iluminada que una superficie que presente una cierta inclinación (como pasa con la Tierra en invierno). Una superficie, por su parte, que esté vuelta de espaldas a la luz, no se verá iluminada en absoluto.

Como un objeto está descompuesto en polígonos, y cada polígono tiene una inclinación distinta, el sistema calculará por separado la iluminación de cada polígono.

---

1 A lo sumo, se recurre a técnicas de dithering para ajustar el color del polígono. El dithering consiste en alternar puntos de distinto color (o distinto nivel de brillo) para conseguir un color (o nivel de brillo) intermedio. Por ejemplo, alternando en un polígono los puntos de color rojo vivo con los de color rojo apagado, se consigue que el color "aparente" del polígono sea de un rojo intermedio.

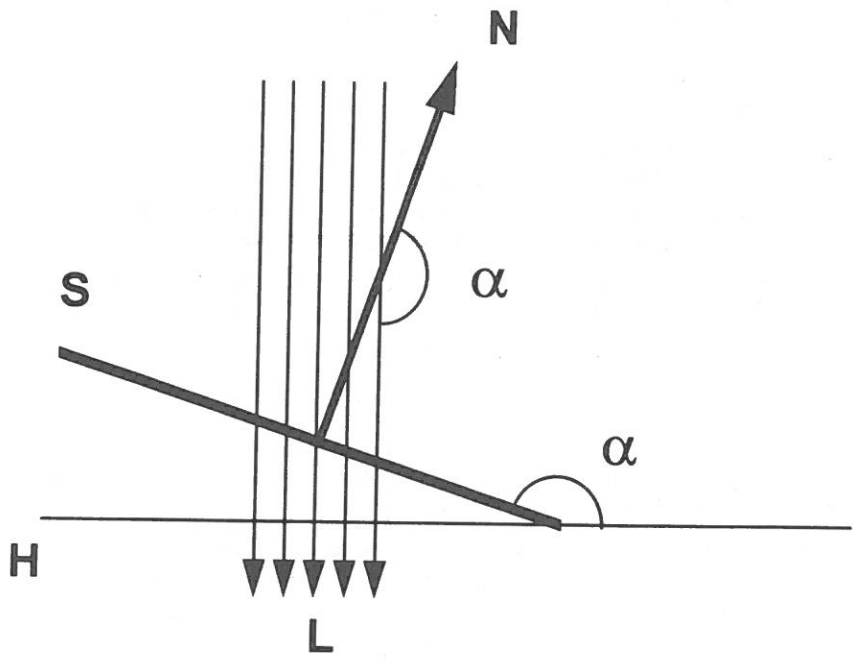

Figura 5.5. Iluminación de una superficie

Vamos a suponer, inicialmente, que existe un único foco de luz y que éste se halla situado a una distancia tan grande que ilumina a todos los objetos desde el mismo ángulo (es decir, los rayos de luz son todos paralelos). La figura 5.5 muestra esta situación en que una serie de rayos de luz paralelos, L, ilumina una superficie S. El vector N es el vector normal a la superficie, mientras que H representa una superficie perpendicular a los haces de luz. Obsérvese que el ángulo que forman S y H es idéntico al que forman L y N.

La cantidad de luz *dirigida* (por unidad de superficie) que incidirá en S, vendrá dada por la fórmula:

$$L_{efectiva} = -L \cdot \cos(\alpha) - L \cdot \cos(L,N)$$

siempre y cuando el ángulo que formen L y N sea mayor de 90º. En caso contrario, la superficie estará situada de espaldas a la luz, y no recibirá iluminación alguna. Obsérvese que, si la superficie está situada de cara a la luz, el cos α es negativo (por ser el ángulo superior a 90º), por lo que el nivel de iluminación es positivo.

La cantidad de luz *total* por unidad de superficie será

$$L_{total} = -L_I \cdot \cos(L_I, N) + L_A$$

si el ángulo que forman los rayos de luz y N es superior a 90°, o

$$L_{total} = L_A$$

en caso contrario, siendo L_I la luz incidente procedente del foco y L_A la luz ambiente.

Obsérvese la importancia que tiene el considerar la luz ambiente. Si no la consideráramos, las partes de los objetos situadas de espaldas a la luz no recibirían iluminación alguna y, a la hora de dibujarlas en la pantalla, aparecerían completamente negras. La luz ambiente nos asegura que cualquier punto de un objeto tendrá un mínimo de iluminación.

El siguiente paso consiste en ver cómo reflejan los puntos del polígono la luz que les llega. Para ello, se definen los llamados **coeficientes de reflexión**, que nos indican qué proporción de la luz incidente reflejan los objetos:

$$L_R = K_R \cdot [-L_I \cdot \cos(N, L_I) + L_A]$$

donde L_R es la cantidad de luz reflejada por unidad de superficie y K_R es el coeficiente de reflexión (en caso de que la superficie esté de espaldas a la luz, desaparecería el término correspondiente a la luz dirigida).

En esta ecuación estamos haciendo dos grandes simplificaciones. En primer lugar, estamos considerando que el objeto refleja por igual la luz ambiente y la luz dirigida, por lo que introducimos un único coeficiente de reflexión. La segunda simplificación, más importante, es que vamos a considerar que el objeto refleja la luz en todas direcciones por igual. En el mundo real, los objetos reflejan parte de la luz en todas las direcciones (fenómeno que se conoce como reflexión difusa), pero otra parte la reflejan más o menos con un ángulo que es igual al de la luz incidente, pero de sentido contrario (como sucede en un espejo). Qué proporción reflejen de una u otra forma es algo que depende del material (brillo). Pero, de nuevo, los cálculos resultantes son demasiado complejos como para tener en cuenta este tipo de iluminación en la mayoría de las aplicaciones de tiempo real. Nosotros vamos a ignorar la reflexión no difusa.

Con la ecuación anterior, tenemos un método de calcular la cantidad de luz que refleja cada punto de un polígono. El último paso consistirá en ajustar el color del punto de acuerdo con dicha cantidad de luz.

Los pasos del proceso de cálculo del color de un cierto polígono serían, con este modelo:

1) Hallar el ángulo que forman la normal al polígono y la luz incidente.
2) Hallar la cantidad total de luz reflejada, teniendo en cuenta el coeficiente de reflexión.
3) Determinar el color resultante según la cantidad de luz reflejada.
4) Dar a todos los puntos del polígono el mismo color.

Obsérvese que, con este modelo, es necesario conocer cuál es el vector normal a cada polígono y cuál el coeficiente de reflexión. El vector normal al polígono es sencillo de hallar conociendo los vértices del mismo, pero suele precomputarse para ahorrar cálculos en tiempo de ejecución. En cuanto al coeficiente de reflexión, puede ser el mismo para todo el objeto,

o distinto para cada uno de sus polígonos. En cualquier caso, es una característica más de los objetos, que habrá que determinar a la hora de diseñarlos gráficamente.

El modelo de sombreado plano es más realista que el de sombreado aleatorio. No sólo continúa el usuario pudiendo percibir la estructura de los objetos, sino que las partes más oscuras de los objetos están donde tienen que estar (según la posición de las fuentes de luz), y ello proporciona una clave más de percepción de profundidad al cerebro, como ya hemos explicado en otro capítulo.

Sin embargo, la simplicidad del sombreado plano hace que este modelo adolezca de imperfecciones: el usuario percibe la forma de los objetos a través de la estructura de polígonos, con lo que todos los objetos aparecen plagados de aristas en los puntos en que un polígono encaja con otro.

El efecto que esto produce es bastante pobre, especialmente cuando los objetos tienen superficies que (idealmente) deberían ser curvas. El efecto es tanto mayor cuantos menos polígonos empleemos para definir al objeto: si definimos una esfera con ocho polígonos, lo que el usuario verá es un octaedro. Si usamos un par de cientos de polígonos para definir la esfera, el resultado final mejorará. Pero, a pesar de todo, el usuario continuará viendo el objeto como "no curvo". La figura 5.2 muestra el efecto del sombreado plano en una esfera.

5.3.3. Sombreado de Gouraud

El modelo de sombreado de Gouraud trata, precisamente, de evitar este efecto de percepción poligonal. Para ello, en lugar de calcular el color de un polígono completo, el algoritmo de Gouraud calcula el color punto a punto, con lo que el resultado final es un objeto en el que el color varía de forma suave de un punto a otro.

Para ello, primero se calcula por separado cuál es el color de cada uno de los vértices del polígono, y luego se asigna un valor intermedio de color a cada punto del polígono, dependiendo de su distancia a cada vértice. Los pasos serían:

1) Hallar el ángulo que forman la normal al **primer vértice** y la luz incidente.
2) Hallar la cantidad total de luz reflejada, teniendo en cuenta el coeficiente de reflexión.
3) Determinar el color resultante **del vértice** según la cantidad de luz reflejada.
4) Repetir los pasos 1 a 3 para cada uno de los otros vértices.
5) Hallar el color de cada punto del polígono interpolando linealmente los valores de color de los vértices.

Obsérvese que, ahora, no se emplea para nada el vector normal al polígono, sino los vectores normales a los vértices (ver figura 5.6), es decir, los vectores perpendiculares al objeto que pasan por los vértices. Dichos vectores tienen que ser definidos a la hora de realizar el diseño gráfico del objeto. Una forma posible (porque no es la única) consiste en definir el vector normal a un vértice como la media de los vectores normales a todos los polígonos de los que el vértice forma parte. La figura 5.6 muestra una versión bidimensional del problema. En ella, V1 es la media de Pa y Pb, mientras que V2 es la media de Pb y Pc.

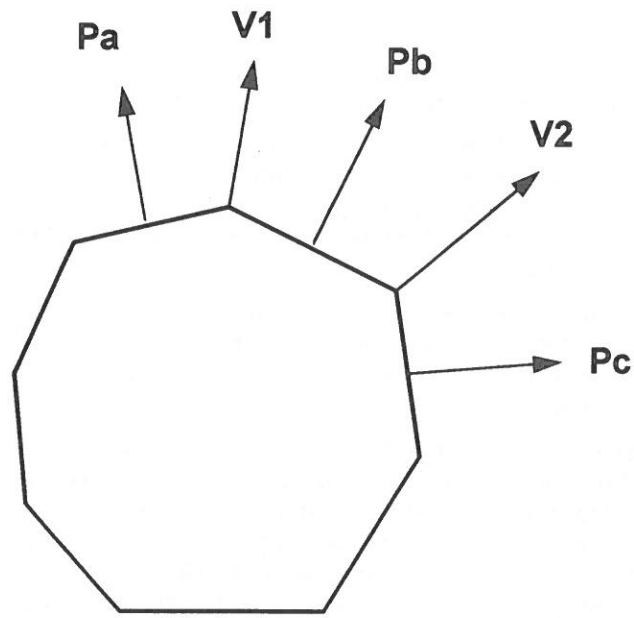

Figura 5.6. Vectores normales a los vértices

Con el modelo de sombreado de Gouraud se resuelve (casi por completo) el problema de la representación de superficies curvas. Al calcularse el color punto a punto, se pierde la diferenciación abrupta entre polígonos, y los objetos con curvas aparecen de una forma mucho más realista. Decimos que se resuelve casi por completo porque, cuando el número de polígonos es bajo, se siguen percibiendo discontinuidades de matiz en las aristas de los polígonos. Pero dichas discontinuidades no son tan aparentes como en el caso del sombreado plano y, además, son indistinguibles cuando se emplea un número suficientemente grande de polígonos.

La figura 5.7 muestra la diferencia en el aspecto final de un objeto, según cuál de los dos métodos de sombreado hasta ahora explicados se utilice.

Dado el realismo que este método permite conseguir en los sombreados, éste es el modelo de iluminación empleado en la mayoría de las aplicaciones de carácter profesional.

El número de cálculos que este método requiere es, obviamente, mayor que el del sombreado plano. En lugar de tener que calcular la luz reflejada una vez por cada polígono del objeto, hay que hacerlo una vez por cada vértice, y realizar una interpolación por cada punto del polígono. Los sistemas comerciales de tratamiento de gráficos 3D y aplicaciones de RV suelen incluir dispositivos hardware especializados en realizar las operaciones de interpolación de color, para así implementar de manera rápida el algoritmo de Gouraud. A título orientativo, podemos decir que el pasar de sombreado plano a sombreado de Gouraud

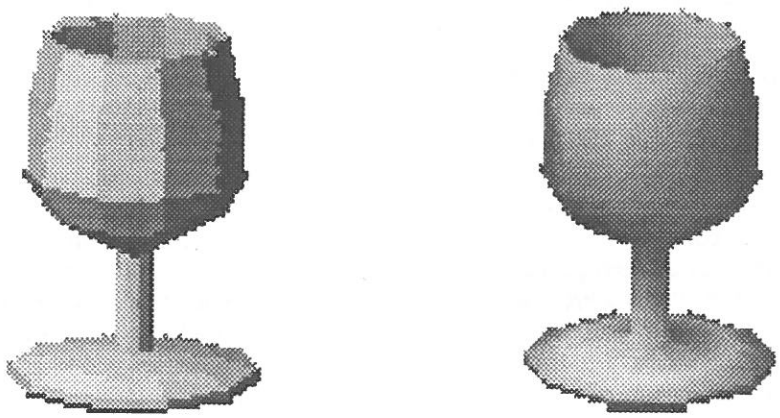

Figura 5.7. Diferencia que se obtiene utilizando sombreado de Gouraud

requiere aumentar entre un 20% y un 50% la capacidad de procesamiento gráfico de un sistema.

5.3.4. Sombreado de Phong

Si se quiere eliminar del todo el problema de las discontinuidades en objetos curvos, puede mejorarse un poco más el algoritmo, utilizando la técnica de sombreado de Phong. Los pasos son:

1) Hallar el vector normal **a un punto** del polígono, interpolando los vectores normales a los vértices.
2) Hallar el ángulo que forman la normal al **punto** y la luz incidente.
3) Hallar la cantidad total de luz reflejada, teniendo en cuenta el coeficiente de reflexión.
4) Determinar el color resultante **del punto** según la cantidad de luz reflejada.
5) Repetir los pasos 1 a 4 para cada uno de los otros puntos del polígono.

Es decir, en lugar de interpolar los colores de los vértices, se interpolan las normales, y luego se halla el color de cada punto. Este algoritmo consigue mejores resultados que el de Gouraud, pero la cantidad de operaciones trigonométricas requeridas (una por cada punto) lo hace inviable para aplicaciones de tiempo real, por lo que se cita sólo como referencia.

5.3.5. Un modelo real

Lo normal en aplicaciones de RV es, por consiguiente, emplear sombreado plano (para aplicaciones de baja calidad gráfica) o sombreado de Gouraud. Sin embargo, los modelos

de iluminación utilizados suelen ser más complejos de lo que los hemos descrito. Las extensiones más comunes a los modelos descritos son las siguientes:

- **Más de una fuente de luz** en las escenas: si las escenas incorporan más de una fuente de luz dirigida, la iluminación total de un polígono (o de un vértice) será la suma de la iluminación producida por todos los focos.
- **Fuentes de luz puntuales**: si las fuentes de luz empleadas están tan alejadas que pueden considerarse situadas en el infinito, todos los rayos de luz serán paralelos. Si el foco, por el contrario, está situado en un punto determinado del mundo virtual (es decir, si es un foco puntual) los rayos de luz irán en todas direcciones a partir del foco, e incidirán sobre cada superficie con un ángulo distinto. En consecuencia, se hace necesario realizar un cálculo más para poder hallar la iluminación que la superficie recibe. Primero tendremos que calcular la dirección con la que los rayos inciden en la superficie (que en el caso de foco situado en el infinito era constante), y luego podemos obtener el ángulo de incidencia[1].
- **Coeficientes de reflexión dependientes del color**: los objetos reales reflejan la luz de manera distinta según el color de la misma. En consecuencia, un modelo más realista consistiría en definir un coeficiente de reflexión diferente para cada una de las tres componentes de color (rojo, azul y verde) y aplicar tres veces la ecuación de cálculo de L_R, calculando el nivel de iluminación para cada color.

Existen otras muchas técnicas que permiten incrementar el realismo de una imagen, como la técnica de trazado de haces (ray-tracing) o las técnicas de radiosidad. Algunas de ellas se emplean profusamente en los programas de animación profesionales, porque la calidad que requiere, por ejemplo, un anuncio de TV es alta y porque no existe la restricción del tiempo real (el que el ordenador tarde 20 minutos en generar la imagen no es importante si, a cambio, se mejora la calidad suficientemente).

Ninguna de estas técnicas, sin embargo, es susceptible de ser utilizada en tiempo real, por la complejidad de cálculo que conllevan. Como consecuencia, no se emplean (hoy por hoy) en las aplicaciones de RV, por lo que caen fuera del objeto del presente volumen.

5.4. EL PROCESO DE LA SÍNTESIS

Con esto, ya tenemos todos los elementos que necesitamos para entender cómo funciona el proceso de la síntesis. Partimos de que tenemos una lista de los objetos que forman la

1 En el caso de fuentes de luz puntuales, consideramos que la fuente de luz ilumina por igual en todas las direcciones. Un modelo más realista, empleado en animaciones profesionales, consiste en suponer una cierta orientación del foco de luz, el cual iluminará más en unas direcciones que en otras. El cuánto ilumine en cada dirección viene dado por un diagrama de radiación. Este modelo de foco requiere mayor procesamiento y capacidad de memoria, y no suele emplearse en aplicaciones de RV. Tampoco suele considerarse, por el mismo motivo, el hecho de que la intensidad de luz decrece con la distancia al foco.

escena, cada uno con su posición y con su definición gráfica, la cual vendrá dada como un conjunto de polígonos.

Los pasos del proceso de síntesis de la imagen son los siguientes:

1) **Transformación de coordenadas:** se calcula la posición de los objetos con respecto al observador.
2) **Clipping:** se eliminan aquéllos objetos o partes de objetos que caen fuera del ángulo de visión del observador.
3) **Cálculo de la iluminación:** se determina cuál es el grado de iluminación de cada polígono o de cada vértice, según vayamos a utilizar sombreado plano o de Gouraud.
4) **Proyección:** se proyectan sobre la pantalla los distintos polígonos que componen cada objeto.
5) **Z-buffering:** se eliminan las partes de los polígonos que están ocultas por otros polígonos situados más cerca del observador.
6) **Sombreado:** se calcula el color de los puntos visibles de cada polígono, y se da dicho valor al pixel de la pantalla correspondiente.
7) **Presentación de la imagen:** se muestra la imagen completa en pantalla, una vez finalizados los cálculos.

Los cuatro primeros pasos forman lo que se conoce con el nombre de etapa de **procesamiento geométrico**, donde el sistema trabaja fundamentalmente a nivel de polígonos. Los pasos 5 y 6, por su parte, son la etapa de **procesamiento ráster**, o rasterización, y en ella el sistema trabaja a nivel de pixels de la pantalla. El último paso, el de presentación, no requiere ningún tipo de procesamiento.

5.4.1. Transformación de coordenadas

Antes hemos visto que los objetos se definían mediante una serie de polígonos, y que éstos se expresaban dando las coordenadas de sus vértices con respecto al centro del objeto. También vimos que cada objeto tenía definida una posición con respecto al centro de la escena.

Para poder mostrar un objeto en la pantalla, lo primero que tendremos que hacer es calcular cómo se ve el objeto desde el punto de vista del observador. En otras palabras, tenemos que calcular cuál es la posición de cada uno de los polígonos del objeto en el **sistema de coordenadas del observador**.

Para ello, no tenemos más que aplicar dos transformaciones seguidas:

- Primero calculamos la posición de cada polígono con respecto al centro de la escena.
- Después calculamos la posición de cada polígono con respecto al observador

La primera de las transformaciones la calculamos a partir de dos datos conocidos: la posición del polígono con respecto al centro del objeto y la posición del objeto con respecto al centro de la escena. El resultado será la posición del polígono con respecto al centro de la escena.

La segunda transformación se calcula a partir del resultado anterior y de la posición del observador dentro de la escena, que el sistema conoce en todo momento. El resultado será la posición del polígono con respecto al observador.

En palabras más técnicas, lo que hacemos es efectuar un par de transformaciones de coordenadas. En la primera, pasamos del sistema de coordenadas del objeto al sistema de coordenadas de la escena. En la segunda, pasamos del sistema de coordenadas de la escena al sistema de coordenadas del observador.

En la práctica, ambos pasos se realizan de una sola vez. Cada una de las transformaciones de coordenadas lleva asociada una matriz, y basta con multiplicar ambas matrices para obtener las ecuaciones del cambio de coordenadas resultante de componer ambas transformaciones.

5.4.2. Clipping

Una vez que tenemos definida la posición de todos los objetos con respecto al observador, podemos eliminar todos aquéllos que caigan fuera del campo de visión del mismo, con el fin de ahorrarnos cálculos posteriores.

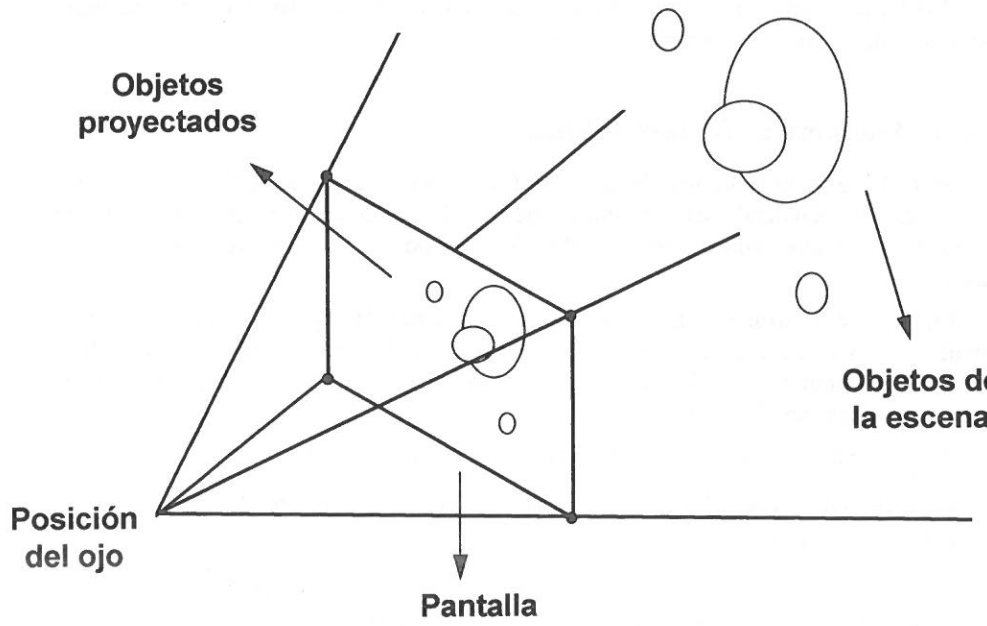

Figura 5.8. Pirámide de visión y plano de pantalla

El campo de visión del observador queda definido por una pirámide de cuatro lados, cuyo vértice está en su ojo, y cuyas cuatro aristas pasan por las esquinas de la pantalla (ver figura 5.8). Dicha pirámide recibe el nombre de **pirámide de visión.**

No sólo podemos eliminar objetos completos. También es posible que haya objetos que caigan parcialmente fuera del campo de visión. En ese caso, lo que se hace es "recortar" los polígonos del objeto, eliminando los vértices que el observador no puede ver y añadiendo los nuevos vértices definidos por el punto donde cada arista del polígono intersecta a los lados de la pirámide.

5.4.3. Cálculo de la iluminación

En esta fase se preparan los datos necesarios para poder calcular luego el color de cada pixel.

En caso de usarse un modelo de iluminación con sombreado plano, se calcula el color de cada polígono teniendo en cuenta el ángulo que forman la luz incidente y el vector normal al polígono, y dicho valor se almacena para su uso posterior.

En caso de usarse sombreado de Gouraud, se calcula el color de cada vértice teniendo en cuenta el ángulo que forman la luz incidente y el vector normal al vértice. Los valores así obtenidos para los distintos vértices se almacenan para su uso en el cálculo del sombreado.

5.4.4. Proyección

En esta etapa, se proyectan los objetos 3D sobre la pantalla. Se trata de una proyección planar (es decir, sobre un plano) y en perspectiva (es decir, las líneas de proyección convergen en un cierto punto del espacio, en lugar de ser paralelas). Por eso el ángulo de visión quedaba definido como una pirámide. Para especificar una proyección de este tipo, basta con indicar la distancia de la pantalla al punto focal (es decir, al ojo del usuario), y la altura y anchura que la pantalla tiene.

Antes de la proyección, cada vértice de un polígono está definido por tres coordenadas X, Y y Z. X e Y representan las coordenadas horizontal y vertical, mientras que Z nos da la profundidad (hacia dentro de la pantalla). Después de la proyección, cada polígono en el espacio 3D queda convertido en otro polígono sobre la pantalla, pero ahora cada vértice queda definido por dos coordenadas (los valores X' e Y' del pixel correspondiente). Las nuevas coordenadas X' e Y' serán diferentes de las originales X e Y, por efecto de la perspectiva. En cuanto a la tercera coordenada, Z, desaparece por efecto de la proyección.

Sin embargo, el sistema no borra la Z original de los vértices, porque le hará falta en el siguiente paso del algoritmo de síntesis.

Hay que hacer notar que la proyección de una escena puede efectuarse sobre la pantalla completa, o sobre una ventana de la misma. En los sistemas inmersivos, la proyección ocupará normalmente toda la pantalla, pero nada impide que en un sistema de sobremesa la escena 3D sólo se muestre en una ventana, dejando el resto de la pantalla para otro tipo de información.

5.4.5. Z-buffering

Una vez que hemos proyectado todos los polígonos, termina la fase de procesamiento geométrico y comienza la rasterización. A partir de aquí, el sistema comienza a operar sobre pixels individuales de la pantalla, y no sobre polígonos completos.

El primer paso consiste en averiguar qué polígonos se ven y cuáles no. Hasta ahora hemos trabajado con los polígonos como si el observador pudiera ver todos los que caen en su campo de visión. Pero lo cierto es que un polígono puede quedar totalmente oculto por otros situados más cerca del observador. O también puede suceder que un polígono oculte *parte* de otro.

En consecuencia, para cada pixel de un polígono proyectado, el sistema calcula si hay otro pixel (de otro polígono) situado más cerca. Si es así, entonces ese punto del polígono no es visible para el observador, y no tenemos que hacer ningún cálculo adicional. En caso de que no hubiera ningún otro pixel más cercano, habrá que realizar los cálculos de sombreado para ese pixel.

¿Cómo calcula el sistema si hay otro pixel más cercano? Utilizando la técnica denominada **z-buffering**. Para comprender cómo funciona dicha técnica vamos a examinar un ejemplo. Supongamos que debemos hallar el menor de los números de una cierta lista:

13 41 23 9 12 11 6 18

Un algoritmo muy sencillo, y fácilmente implementable, sería ir recorriendo la lista de izquierda a derecha, anotando en una casilla el menor de los números que hasta el momento hayamos revisado. Al terminar de recorrer la lista, la casilla contendrá el menor de todos los números de la lista. Paso a paso, el sistema sería el siguiente (en cada paso, indicamos con negrita cuál es el último número revisado):

PASO 1 Anoto en la casilla el primero de los números, en este caso el 13.

Lista **13** 41 23 9 12 11 6 18

Mínimo 13

PASO 2 Compruebo si el siguiente número es mayor o menor que el mínimo ya almacenado. Como 41 es mayor que 13, dejo la casilla del mínimo tal como está.

Lista 13 **41** 23 9 12 11 6 18

Mínimo 13

PASO 3 23 es mayor que 13, luego dejo la casilla del mínimo tal como está.

Lista 13 41 **23** 9 12 11 6 18

Mínimo 13

PASO 4 9 es menor que 13, luego escribo 9 en la casilla del mínimo

Lista 13 41 23 **9** 12 11 6 18

Mínimo 9

PASO 5 Lista13 41 23 9 **12** 11 6 18
Mínimo 9

PASO 6 Lista13 41 23 9 12 **11** 6 18
Mínimo 9

PASO 7 Lista13 41 23 9 12 11 **6** 18
Mínimo 6

PASO 8 Lista13 41 23 9 12 11 6 **18**
Mínimo 6

De esta manera tan simple, podemos calcular siempre el mínimo de una lista, independientemente del orden en que los números se presenten o de lo larga que la lista sea.

Pues bien, este es el principio en que se basa la técnica del z-buffering. Supongamos que hay varios polígonos en la escena que se superponen unos a otros. Como resultado, habrá pixels de la pantalla donde quedarán proyectados varios puntos de polígonos diferentes. En la figura 5.9, por ejemplo, sobre el pixel P se proyectarán tres puntos diferentes: uno de cada uno de los polígonos A, B y C.

¿Cuál de los tres puntos proyectados debe dibujar el sistema en el pixel P? Pues, evidentemente, aquél que se encuentre más cerca del observador, porque los otros dos

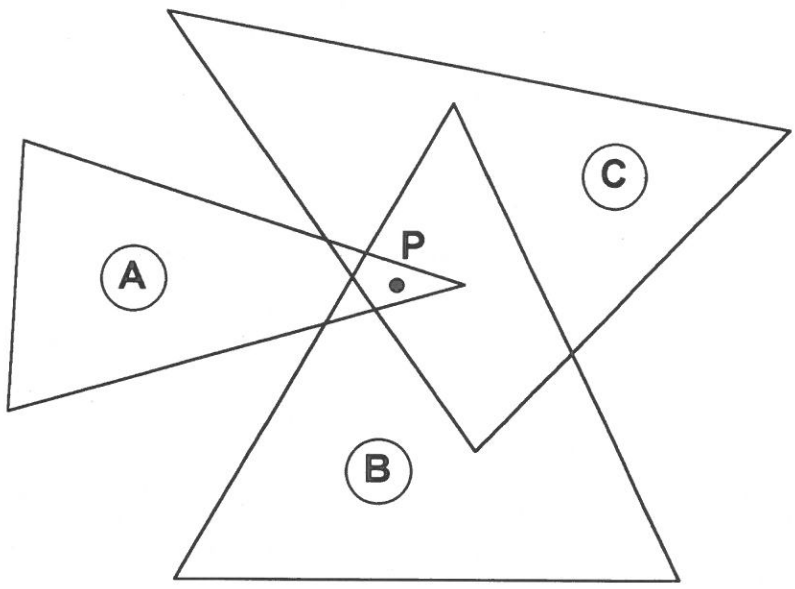

Figura 5.9. Polígonos superpuestos

quedarán tapados por éste. Así, si el polígono A fuera el más cercano en la figura 5.9, en el pixel P habrá que dibujar el punto correspondiente de A, y no el de B o C.

¿Cómo puede saber el sistema cuál de los tres puntos es el más cercano? Pues empleando el algoritmo anteriormente expuesto, porque el punto más cercano será aquél que tenga una menor coordenada de profundidad Z.

En consecuencia, el sistema no tiene más que ir cogiendo un polígono detrás de otro y mantener en una casilla el valor del mínimo Z que hasta el momento haya encontrado. Cada vez que encuentra un Z menor, dibujará en la pantalla el punto correspondiente, y almacenará en la casilla el nuevo valor de Z. De esta forma, al finalizar, el pixel P tendrá dibujado el punto con menor valor de Z.

Esto hay que hacerlo para todos los pixels de la pantalla. Por razones de eficiencia, este algoritmo se lleva a cabo en paralelo para todos los pixels, por lo que necesitaremos una casilla por pixel para guardar el valor de Z mínimo correspondiente. El conjunto de todas esas casillas se denomina **z-buffer**.

El algoritmo completo sería como sigue:

1) Inicializar todas las casillas del z-buffer con el valor máximo que la coordenada de profundidad pueda adoptar.
2) Tomar el primer polígono de la escena (ya proyectado).
3) Tomar el primer punto del polígono.
4) Calcular el valor Z_0 de su coordenada de profundidad
5) Si Z_0 es mayor que el valor contenido en la casilla correspondiente del z-buffer, descartar el punto (porque eso indica que hay otro más cercano). Si Z_0 es menor, dibujar el punto en la pantalla y almacenar el valor Z_0 en la casilla del z-buffer.
6) Repetir los pasos 4 y 5 para los restantes puntos del polígono.
7) Repetir los pasos 3 a 6 para los restantes polígonos de la escena.

Al acabar, cada pixel de la pantalla tendrá dibujado el punto más próximo al observador, de todos los puntos que se proyectaron sobre él.

Los pasos 4 y 5 necesitan una explicación adicional. En el paso 5, el sistema deberá dibujar en la pantalla el punto, en caso de que el valor de Z_0 así lo indique. Lo que el sistema deberá hacer es calcular el color del punto y darle al pixel de la pantalla dicho color. En el apartado siguiente, al hablar del sombreado, explicaremos cómo realizar esta operación.

En cuanto al paso 4, ¿cómo calcula el sistema la coordenada de profundidad de un punto cualquiera de un polígono? Pues aplicando una simple regla de tres. O, si se prefiere la terminología técnica (que queda más vistosa), realizando una **interpolación lineal**.

Al hablar de la proyección, dijimos que el sistema conservaba la coordenada de profundidad (Z) de cada vértice de los polígonos, a pesar de que dicha coordenada perdía sentido al proyectar el polígono. La razón es, precisamente, poder realizar el z-buffering. Conociendo las coordenadas y valores de profundidad de los vértices, y conociendo las coordenadas del punto para el cual quiero calcular Z, la profundidad de este punto se obtendría tal como muestra la figura 5.10.

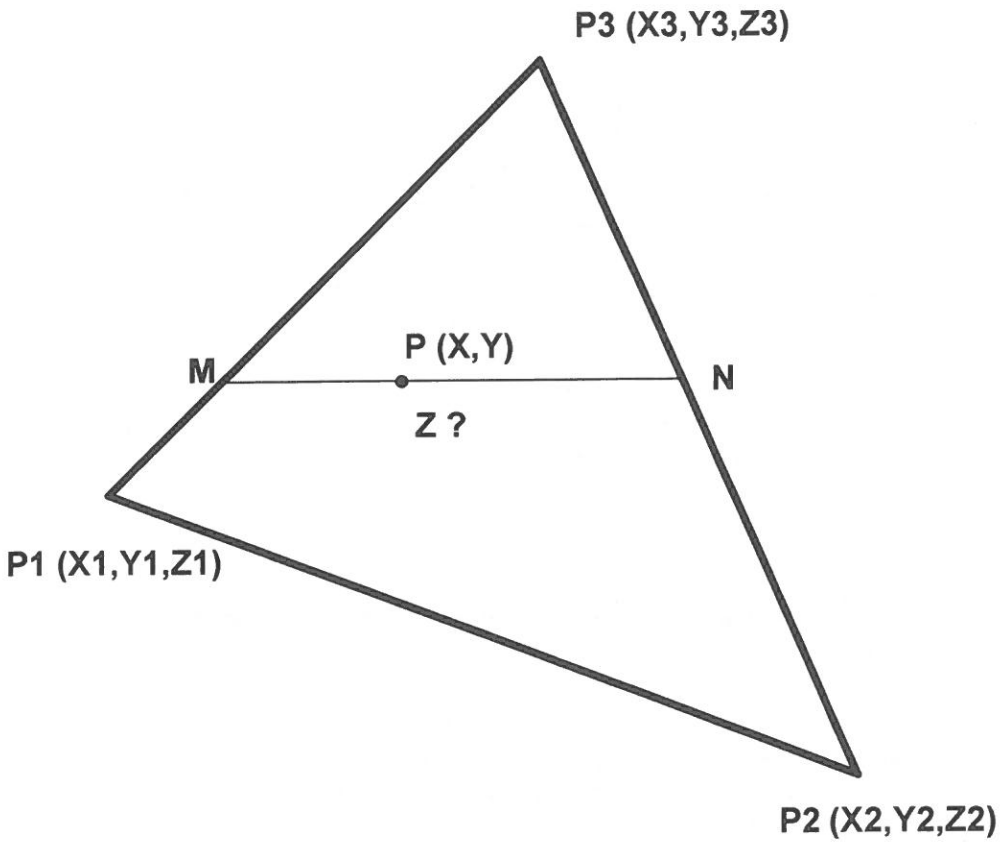

Figura 5.10. Interpolación de los valores de Z

Primero calculamos la Z del punto M mediante una regla de tres:

$$Z_M = \frac{(X_M - X_1)*(Z_2 - Z_1)}{X_2 - X_1}$$

Después hacemos lo mismo con el punto N:

$$Z_N = \frac{(X_N - X_2)*(Z_3 - Z_2)}{X_3 - X_2}$$

Finalmente, calculamos la Z del punto P mediante una última regla de tres, a partir de las de los puntos M y N:

$$Z_p = \frac{(X_p - X_M)*(Z_N - Z_M)}{X_N - X_M}$$

En la práctica, todos estos cálculos se realizan de una sola vez.

Hay que destacar que, matemáticamente hablando, esta forma de calcular la Z de un punto intermedio de un polígono es incorrecta. Al aplicar las fórmulas anteriores, estamos suponiendo que Z varía linealmente en el polígono, lo cual es falso por el efecto de la proyección. Sin embargo, el intentar aplicar las fórmulas exactas es tremendamente costoso desde el punto de vista computacional, y el método expuesto da una aproximación lo suficientemente buena, por lo que todos los sistemas de generación de gráficos 3D en tiempo real emplean el método indicado. El único caso donde la aproximación no es adecuada es cuando se utilizan texturas, como más abajo se expone.

Con la técnica del z-buffering, podemos dibujar cualquier número de polígonos en la pantalla, en cualquier orden, en la seguridad de que la imagen final contendrá sólo aquellos puntos de los polígonos que no estén tapados por otros. No tenemos que preocuparnos de dibujar los polígonos en un orden determinado, porque el algoritmo del z-buffering se encarga de descartar automáticamente aquellas partes de un polígono que quedan ocultas por otro.

5.4.6. Sombreado

En la sección anterior hemos visto que, cuando el sistema encontraba que la coordenada de profundidad Z de un punto era menor que la del z-buffer, dibujaba el punto en la pantalla.

Para poder dibujar el punto, es necesario antes que el sistema calcule el color del mismo. El cómo lo haga va a depender de qué modelo de iluminación se esté utilizando.

En caso de emplearse sombreado plano, todos los puntos del polígono tienen el mismo color, luego el sistema le da al punto el color del polígono al que pertenece (que ya fue calculado y almacenado durante la etapa de cálculo de iluminación).

En el caso de que se esté usando sombreado de Gouraud, el sistema tendrá que realizar algún cálculo extra. En concreto, lo que el sistema hará será interpolar linealmente el color de los vértices del polígono, exactamente igual que con la coordenada Z[1].

En cualquiera de los dos casos, una vez que el sistema tenga el color del punto, dará dicho color al correspondiente pixel de pantalla.

En consecuencia, el algoritmo para la rasterización de un cierto polígono nos quedaría de acuerdo con las figuras 5.11 y 5.12.

5.4.7. Presentación de la imagen

Hasta el momento, hemos estado hablando en los pasos anteriores como si el sistema trabajara directamente sobre la pantalla a la hora de realizar la síntesis de las imágenes. Pero en la realidad no sucede así: el sistema no proyecta los polígonos "sobre la pantalla", ni escribe ningún valor de color en los pixels "de la pantalla". Si lo hiciera, la pantalla se iría

1 Este cálculo de los colores mediante una interpolación no pasa de ser un cálculo aproximado, al igual que sucede con la interpolación de la coordenada Z.

Sombreado plano:

```
para cada pixel del polígono
  calcular su Z, por la interpolación de la Z de los vértices
  si es menor que la almacenada en el z-buffer
    dar al pixel el color correspondiente a ese polígono
    sustituir el valor del z-buffer por Z
  fin
fin
```

Figura 5.11. Algoritmo de rasterización para sombreado plano

Sombreado de Gouraud

```
para cada pixel del polígono
  calcular su Z, por interpolación de la Z de los vértices
  si es menor que la almacenada en el z-buffer
    hallar el color del punto, interpolando el de los vértices
    dar al pixel dicho color
    sustituir el valor del z-buffer por Z
  fin
fin
```

Figura 5.12. Algoritmo de rasterización para sombreado de Gouraud

actualizando poco a poco, a medida que el sistema fuera dibujando polígonos, y el efecto final sería muy pobre.

Lo que sucede en la realidad es que el sistema va escribiendo los pixels sobre una zona de memoria reservada a ese menester. Una vez que se ha terminado de sintetizar la imagen completa, el sistema la presenta en la pantalla de un solo golpe.

Para poder hacer esto, el hardware gráfico de los sistemas de RV tiene un doble buffer de imagen (en inglés, frame buffer). En todo momento los circuitos de video trabajan con sólo uno de los buffers, presentando su contenido en la pantalla. Mientras tanto, el sistema va calculando la siguiente imagen y almacenando los resultados parciales en el otro buffer.

Cuando ha terminado de sintetizar la nueva imagen, el sistema ordena a los circuitos de video que cambien de buffer, y así la nueva imagen aparece en la pantalla. Entonces, comienza a sintetizar la imagen siguiente, utilizando el buffer que acaba de quedar libre.

Esta técnica le resultará bastante familiar a cualquiera que haya trabajando programando con tarjetas gráficas en un PC. En una tarjeta gráfica cualquiera (y dependiendo del modo de video en que nos encontremos), la memoria de pantalla puede estar dividida en una serie de páginas, de las cuales sólo una aparece en pantalla en un determinado momento. Mientras

se está mostrando una página por la pantalla, el ordenador puede ir escribiendo datos en otra página cualquiera. Al acabar, basta con ordenar a la tarjeta que conmute de página para que los nuevos datos se muestren en el monitor.

En caso de que la tarjeta gráfica empleada no tenga doble buffer de imagen, o una memoria de pantalla con múltiples páginas, se puede simular esta técnica haciendo uso de la memoria normal: el sistema puede ir almacenando la imagen en una zona de la memoria normal cualquiera. Al acabar, dicha zona de memoria se copia de un solo golpe sobre la memoria de pantalla, con lo que la nueva imagen aparece. La única desventaja de esta solución es que es algo más lenta, porque la operación de copiado tarda mucho más tiempo que la mera conmutación hardware utilizada por los sistemas con doble buffer de imagen.

En los sistemas de RV estereoscópicos que utilizan multiplexación espacial (es decir, en los sistemas que tienen una pantalla para cada ojo), es importante mantener una sincronización lo más perfecta posible entre las dos imágenes. Una técnica muy utilizada consiste en emplear un doble frame buffer para cada ojo, dibujar ambas imágenes (en paralelo o una a continuación de la otra) y, cuando se ha terminado de generar las dos imágenes, conmutar a la vez ambos frame buffer. De esta manera, el cambio de imagen se realiza simultáneamente para los dos ojos.

5.4.8. El algoritmo de síntesis al completo (versión simple)

Con todo lo anterior, la versión completa de nuestro algoritmo de síntesis simplificado podría ser algo así como lo que indica la figura 5.13.

Como vemos, existen dos partes claramente diferenciadas en el proceso de síntesis de las escenas. En una primera parte, la **fase de procesamiento geométrico,** se efectúa una serie de cálculos y transformaciones sobre los polígonos que componen cada objeto. Esta fase comprende las transformaciones geométricas, el clipping, la iluminación y la proyección, y consta básicamente de una serie de operaciones en punto flotante cuyo número es proporcional al de polígonos existentes en la escena.

Los dos pasos siguientes, z-buffering y sombreado, forman la **fase de procesamiento ráster,** e implican una serie de operaciones en las que el sujeto ya no son los polígonos como un todo, sino los pixels individuales que los componen. Durante esta fase, las operaciones que se efectúan son interpolaciones lineales y trasiego de información hacia el z-buffer y el buffer de imagen. El número de operaciones que se realice durante esta fase no depende del número de polígonos, sino de la superficie que éstos ocupen una vez proyectados en pantalla.

La arquitectura de los sistemas de generación de gráficos 3D suele reflejar esta división del proceso de síntesis de la imagen. Normalmente, un DSP o procesador especializado en la realización de operaciones de punto flotante se encarga de efectuar los cálculos necesarios durante la fase de procesamiento geométrico, mientras que para el procesamiento ráster se suelen incluir circuitos integrados especializados en este tipo de operaciones, generalmente desarrollados a medida.

Esta división también tiene reflejo en las especificaciones de los sistemas. Hay dos magnitudes que es importante considerar a la hora de evaluar un sistema de generación de

```
borrar el buffer donde vayamos a dibujar la nueva imagen;
inicializar el z-buffer;
para cada objeto de la escena{                                          } TRANSFORMACIÓN
  calcular la matriz M de posición con respecto al observador;          } DE COORDENADAS
  para cada polígono del objeto{
    calcular, a partir de M, la posición con respecto al observador;    }
    si el polígono cae fuera del ángulo de visión{                      } CLIPPING
      desechar el polígono;
    }
    en caso contrario{
      si cae parcialmente fuera{
        recortar la parte sobrante;
      }
      para cada vértice del polígono{                                   } ILUMINACIÓN
        calcular su grado de iluminación;
        calcular su color a partir del grado de iluminación;            } PROYECCIÓN
        proyectar el vértice sobre la pantalla;
      }
      para cada pixel del polígono proyectado{                          } Z-BUFFERING
        calcular su Z, por interpolación de la Z de los vértices;
        si es menor que la almacenada en el z-buffer{                   } SOMBREADO
          sustituir el valor del z-buffer por Z;
          hallar el color del punto, interpolando el de los vértices;
          dar al pixel dicho color;
        }
      }
    }
  }
}                                                                       } PRESENTACIÓN
presentar la nueva imagen;
```

Figura 5.13. Algoritmo de síntesis simplificado

gráficos 3-D. Por un lado, los fabricantes suelen dar el número de polígonos por segundo que el sistema es capaz de procesar. Esta cantidad nos habla de la potencia del procesador geométrico empleado. Por otro lado, se suele especificar la tasa de generación de pixels (pixel rate), que nos informa de cuántos pixels es capaz de tratar por segundo el procesador ráster.

Ambas especificaciones son importantes. En una aplicación concreta, el cuello de botella puede estar en el procesador geométrico, mientras que en otra puede ser el procesador ráster el que ponga los límites. Las características del modelo de iluminación que escojamos influirán de manera distinta en uno y otro. Así, el aumentar el número de fuentes de luz hace que aumente la carga del procesador geométrico, pero no altera para nada el proceso de rasterización. De la misma forma, determinadas técnicas como el anti-aliasing (del que más adelante hablaremos) imponen mayores exigencias al procesador ráster, mientras que el procesador geométrico no se verá afectado.

En nuestras aplicaciones, deberemos tener cuidado en seleccionar el subsistema gráfico de forma que se alcance un adecuado balance entre los dos procesos. Si no podemos conseguir

un subsistema que disponga de la relación de procesamiento adecuada, entonces debemos aprovechar las capacidades infrautilizadas.

Por ejemplo, si el procesador geométrico no está al límite de su capacidad, podemos aprovechar para aumentar el número de polígonos utilizados para definir cada objeto (con lo que se obtiene una mejor definición de éstos). Al aumentar el número de polígonos de un objeto, el número medio de pixels por polígono disminuye, y el número total de pixels que el objeto ocupará en pantalla será más o menos el mismo, con lo que apenas aumenta el trabajo de rasterización. Una forma simple de visualizarlo es la siguiente: aunque aumentemos el número de polígonos de un objeto, la proyección de éste en pantalla ocupará más o menos la misma superficie, por lo que el número de pixels correspondiente al objeto (y, por consiguiente, el trabajo de rasterización) no sufre apenas modificación.

Si fuera el procesador ráster el que estuviera infrautilizado, podríamos pensar en aumentar la resolución, utilizando una ventana más grande o una pantalla mejor. Al aumentar la resolución, cada polígono proyectado abarca un mayor número de pixels en pantalla; el procesamiento geométrico no sufre incremento alguno, pero el trabajo de rasterización sufrirá un aumento proporcional al de la resolución.

Finalmente, cada aplicación presenta requerimientos diferentes, por cuanto que no todas exigen la misma calidad de escena, ni todas tratan con el mismo tipo de objetos. Así, existe una diferencia abismal entre generar una escena con unos pocos objetos a los que se aplica un modelo de iluminación de alta calidad y dibujar otra escena con una multitud de objetos usando un modelo de iluminación más pobre. Como consecuencia, el proceso de síntesis debe adaptarse a cada aplicación particular y dimensionar nuestro sistema para obtener un rendimiento máximo.

5.5. OPTIMIZACIÓN DEL PROCESO DE SÍNTESIS

Si queremos aumentar la calidad de las escenas generadas por nuestro sistema de RV, existen dos mecanismos directos hacerlo: aumentar el número de polígonos de los objetos, y aumentar la resolución de la pantalla.

Como hemos visto, aumentar el número de polígonos incide de forma directa en la cantidad de procesamiento necesaria durante la fase de procesamiento geométrico, mientras que aumentar la resolución de la pantalla requiere un incremento en la capacidad de rasterización.

En consecuencia, ambos mecanismos tienen un límite marcado por la propia capacidad de proceso de las distintas partes del sistema. Una vez alcanzado dicho límite, el aspecto gráfico de la aplicación deja de ser mejorable, a menos que...

A menos, claro está, que seamos capaces de optimizar el algoritmo de síntesis para tratar de hacer lo mismo en menos tiempo. De aquí que, en los últimos años, se haya desarrollado un conjunto de técnicas destinado a reducir el tiempo de proceso necesario para sintetizar una imagen.

Como vamos a ver, dichas técnicas están basadas, fundamentalmente, en el sentido común. Puesto que el tiempo de proceso es un bien escaso, es necesario sacar el máximo

provecho del tiempo dedicado a la síntesis. Y "sacar el máximo provecho" se traduce en dos principios fundamentales:

PRINCIPIO 1: Si el hacer algo no sirve para nada, no lo haga.

En una escena determinada, por ejemplo, los objetos pueden estar situados en cualquier posición, tanto delante como detrás del observador. Evidentemente, todas las operaciones que realicemos con objetos situados detrás son tiempo perdido, porque no van a salir en la imagen final. En consecuencia, parece lógico tratar de eliminar dichos objetos lo antes posible dentro del proceso de la síntesis. Esta técnica se conoce con el nombre de **selección**. Además de la de selección, en este apartado expondremos otras técnicas destinadas a reducir el tiempo de síntesis por el procedimiento de eliminar operaciones inútiles.

PRINCIPIO 2: Dedique más recursos a lo que más productivo sea

Suponga una escena en la que hay dos objetos iguales delante del observador, uno situado muy próximo y el otro muy alejado. Cuando proyectemos sobre la pantalla ambos objetos, el primero ocupará una zona de pantalla relativamente grande, y el usuario será capaz de percibir un razonable nivel de detalle. El objeto alejado, sin embargo, ocupará una zona pequeña de la pantalla, y es posible que la mayoría de los detalles del objeto sean imposibles de percibir. En estas circunstancias, ¿tiene sentido dedicar el mismo tiempo de procesamiento geométrico a un objeto que al otro? Parece obvio que la respuesta es no, y a solucionar este problema están dedicadas las llamadas **técnicas de variación dinámica del nivel de detalle**, de las que más abajo hablaremos.

En este apartado vamos a exponer un conjunto de técnicas destinadas a reducir el tiempo requerido para la generación de las imágenes. Algunas de ellas conducen a modificaciones del algoritmo de síntesis, mientras que otras se basan en la modificación inteligente del conjunto de datos de entrada que el proceso utiliza.

5.5.1. Selección (culling)

A la hora de efectuar la síntesis de una escena 3D, muchos de los objetos de la misma (cerca de la mitad, por término medio) no van a poder aparecer en la imagen final, por estar situados completamente fuera de la pirámide de visión del usuario. En consecuencia, es una pérdida de tiempo determinar la posición y realizar el clipping de cada polígono individual del objeto.

Si fuéramos capaces de eliminar dichos objetos al principio del proceso de la síntesis, podríamos reducir a casi la mitad el tiempo necesario para sintetizar una escena de una complejidad determinada.

La mayoría de los objetos situados fuera de la pirámide de visión del usuario pueden directamente eliminarse al principio del proceso de síntesis, sin más que efectuar una sencilla comparación entre la posición del objeto y la posición y orientación de la cámara.

La manera de hacerlo consiste en emplear los denominados **volúmenes de recubrimiento**. Un volumen de recubrimiento es una figura geométrica definida de tal manera que todos los polígonos del objeto caen dentro de la misma. Por ejemplo, podemos definir alrededor del objeto una esfera de un cierto radio que contenga al objeto por entero.

Suponiendo que utilizamos esferas como volúmenes de recubrimiento, la manera de emplearlas sería la siguiente: a la hora de efectuar la síntesis de la escena, comprobamos primero si el objeto se halla a espaldas de la cámara y, en caso afirmativo, si la distancia del objeto al plano de la cámara es mayor que el radio de la esfera de recubrimiento (ver figura 5.14). Si es así, entonces podemos olvidarnos del objeto y pasar al siguiente, porque todos los polígonos componentes caen fuera de la pirámide de visión.

Con esta simple técnica nos ahorramos, de forma muy sencilla, el procesamiento de casi la mitad de los objetos de la escena, por término medio. Podemos usar otro tipo de volumen de recubrimiento (un cilindro o un paralelepípedo, por ejemplo). Cuanto más "ajustado" esté el volumen de recubrimiento a la forma del objeto, más objetos podremos eliminar; pero no conviene utilizar volúmenes muy complejos, porque de lo que se trata es de eliminar los objetos con una comprobación lo más simple posible.

Evidentemente, la técnica puede llevarse más allá y eliminar no sólo los objetos que quedan a espaldas de la cámara sino, en general, todos los que quedan fuera de la pirámide

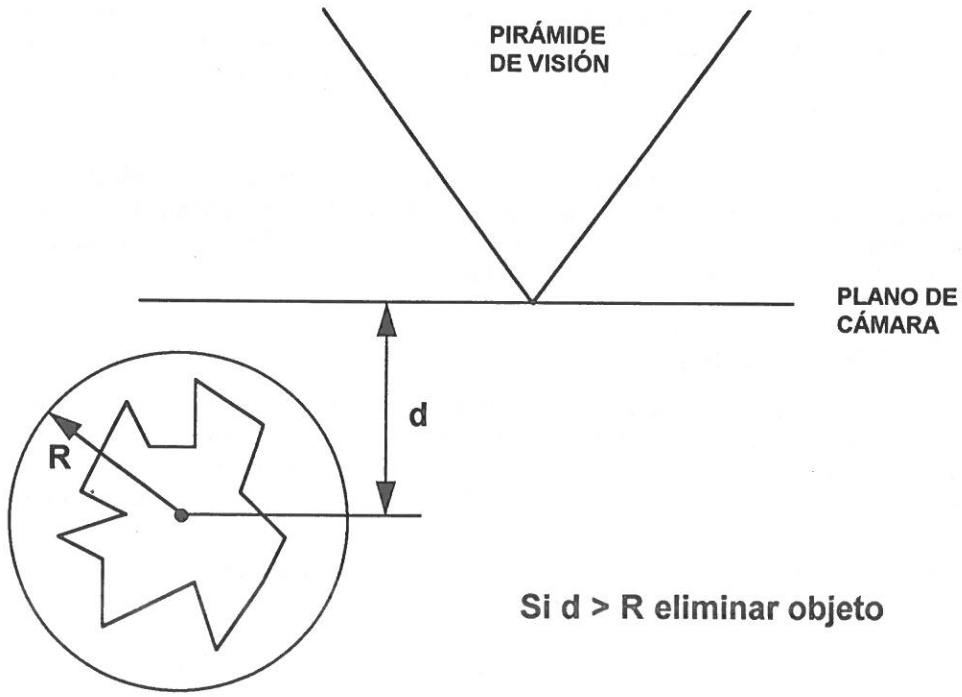

Figura 5.14. Ejemplo de selección

de visión. Utilizando las mismas esferas de recubrimiento, los cálculos para los objetos situados por delante del plano de la cámara son algo más complejos, pero no excesivamente. Indudablemente, el ahorro en el procesamiento de polígonos compensa más que de sobra esos pocos cálculos adicionales.

Para el caso de grandes objetos o superficies, con los que no puede usarse el enfoque de las esferas de recubrimiento, existen otros enfoque distintos (basados por ejemplo en dividir en retículas una determinada región y organizar la retícula jerárquicamente), pero el concepto es el mismo: tratar de eliminar del proceso de síntesis lo antes posible aquellas partes del mundo virtual que, por su posición, no van a poder aparecer de todas formas en la imagen.

Existe un caso particular en el que se puede optimizar aún más el proceso de selección, recurriendo a las llamadas **técnicas de partición espacial**. Dicho caso es cuando una escena contiene una gran cantidad de información estática (por ejemplo, si hay muchos objetos inmóviles o si hay unos pocos objetos inmóviles pero con un gran número de polígonos cada uno).

En este caso, se puede dividir la escena en sectores, y mantener una lista de qué objetos estáticos caen total o parcialmente en cada sector. A la hora de sintetizar la imagen, sólo

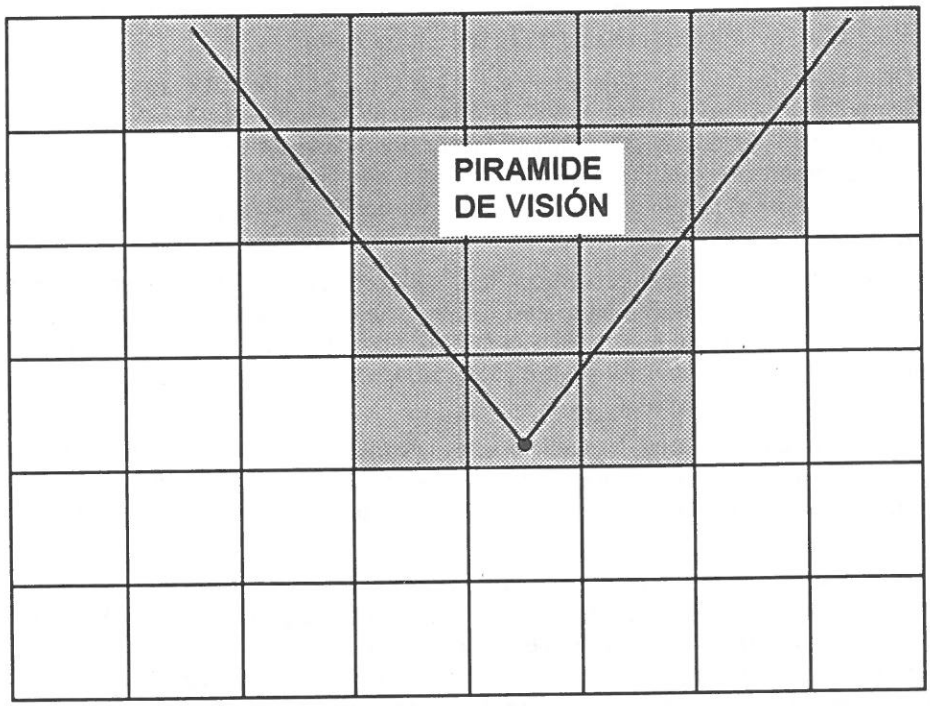

Figura 5.15. Selección por división en sectores

procesaremos aquellos objetos que estén dentro de uno de los sectores intersectados por la pirámide de visión. En la figura 5.15, sólo se procesarán los objetos que estén dentro de (al menos) uno de los sectores marcados.

Obsérvese que las técnicas de selección disminuyen el tiempo de procesamiento geométrico, pero no el tiempo de rasterización necesario, porque los polígonos de los objetos eliminados mediante estas técnicas hubieran sido eliminados de todas las maneras durante la fase de clipping.

5.5.2. Eliminación de caras ocultas

Ya hemos visto anteriormente, al hablar de las técnicas de sombreado, cómo los vectores normales a los polígonos de un objeto se empleaban para los cálculos de iluminación. Pero dichos vectores normales presentan otra utilidad adicional, que nos va a permitir reducir aún más el tiempo de síntesis de una escena.

Dado un objeto cualquiera dentro de nuestra escena tridimensional, existen dos maneras de definir el vector normal a cada uno de sus polígonos: hacia "fuera" o hacia "dentro" del objeto. La convención habitual es definir los vectores normales hacia fuera de los objetos (se podría adoptar la convención inversa si se quisiera; lo importante es que los vectores normales de todos los polígonos se definan del mismo modo).

Observe la figura 5.16. Dado un objeto y una cierta posición del observador, algunos polígonos del objeto estarán "de frente" al observador, mientras que otros quedarán en la parte "posterior" del objeto. Obviamente, los polígonos situados en la parte posterior (con respecto al observador) no serán visibles una vez que la escena haya terminado de ser sintetizada, porque el propio volumen del objeto los oculta. En la figura, los polígonos que el observador podrá ver aparecen marcados en trazo más grueso.

En nuestro algoritmo de síntesis simplificado, los polígonos situados en la parte posterior de los objetos son procesados hasta casi el final. Los puntos correspondientes sólo son eliminados una vez que se ha llegado a la fase de z-buffering. En ese momento, todos los puntos pertenecientes a polígonos ocultos quedarán tapados por otros puntos de los polígonos situados de frente al observador.

Está claro que resulta un desperdicio tener que efectuar tantas operaciones para estos polígonos. La técnica de eliminación de las caras ocultas trata precisamente de evitar este hecho, empleando los vectores normales de los polígonos para eliminarlos antes del proceso de clipping.

Si estudiamos de nuevo la figura 5.16, veremos que existe una forma muy simple de eliminar casi todos los polígonos ocultos. Todos los polígonos situados de frente al observador tienen un vector normal que apunta "hacia el observador", es decir, el ángulo que forman el vector normal y el vector de posición del polígono es superior a 90º. Esto es lo que sucede, por ejemplo, con las caras A y D en la figura. Los polígonos cuya normal forma un ángulo menor de 90º con el vector de posición (como es el caso de la cara C) están todos situados en la parte posterior del objeto.

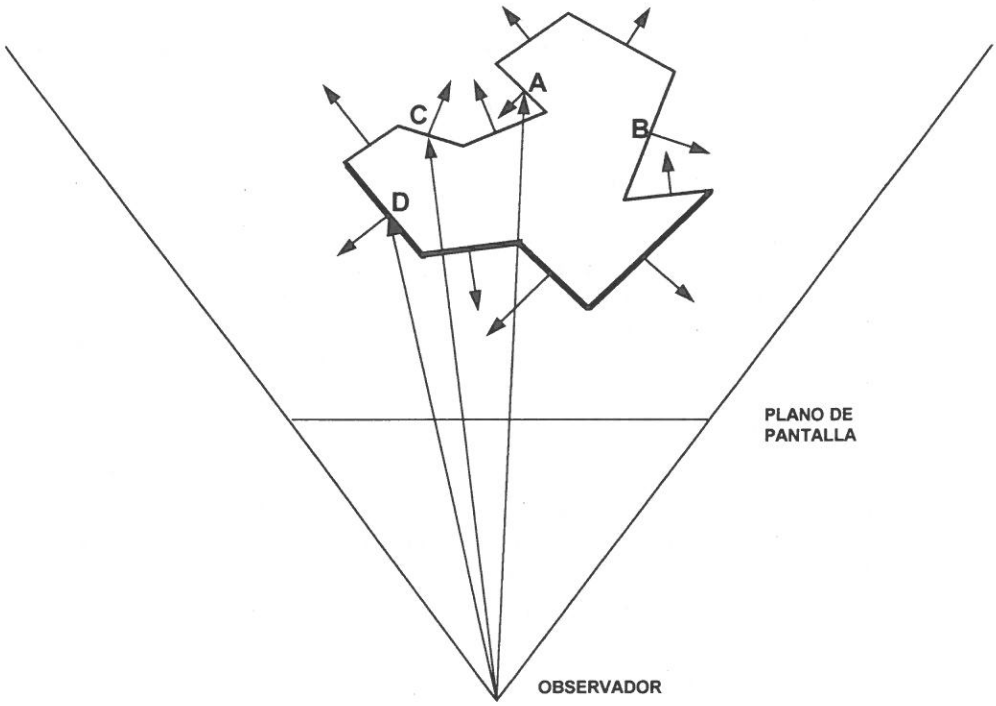

Figura 5.16. Identificación de caras ocultas

En consecuencia, para que un polígono pueda verse en la imagen final, es condición necesaria que el ángulo entre la normal y el vector de posición sea mayor de 90º, por lo que podemos eliminar directamente todos aquellos polígonos que no cumplan esta condición. Como vector de posición del polígono podemos tomar el vector de posición de cualquiera de sus vértices. En nuestro ejemplo, realizando esta pequeña comprobación nos desharíamos de todos los polígonos situados en la parte posterior del objeto, excepto los marcados como A y B.

Con esta sencilla técnica podemos eliminar en una fase temprana del proceso de síntesis un 50% de los polígonos (por término medio). La técnica de eliminación de caras ocultas permitirá disminuir tanto el tiempo de procesamiento geométrico como el tiempo de rasterización, porque nos ahorramos los pasos de clipping, iluminación, proyección, z-buffering y sombreado de los polígonos afectados.

5.5.3. Texturado

Uno de los problemas que presentan las imágenes generadas mediante nuestro algoritmo de síntesis simplificado es el hecho de que los polígonos son absoluta y decididamente lisos.

Los cálculos de sombreado permiten obtener gradaciones de color e intensidad entre uno y otro vértice, pero la apariencia de cada polígono es la de una superficie lisa.

Si tenemos que representar un objeto de superficie muy irregular o rugosa, la única solución consiste en emplear un gran número de polígonos para el objeto, con el fin de aproximar lo más posible dicha superficie.

Supongamos, por ejemplo, que nuestra escena contiene una pared de ladrillo. Podemos representar esa pared mediante un único polígono de gran tamaño, lo que simplificaría mucho los cálculos geométricos, pero la pared será entonces completamente lisa. O podemos emplear múltiples polígonos para definir los ladrillos individuales que componen la pared, en cuyo caso parecerá realmente una pared de ladrillo, a costa de una cantidad considerable de procesamiento geométrico.

Este procesamiento geométrico adicional se debe a que hemos convertido cada ladrillo individual de la pared en un objeto en el espacio 3D. Si hubiera alguna razón relacionada con la lógica de la aplicación para que los ladrillos fueran objetos 3-D (por ejemplo, que el usuario pudiera atisbar entre las juntas buscando algún objeto diminuto), nada habría que oponer a este proceder. Sin embargo, en nuestro caso hemos convertido los ladrillos en objetos tridimensionales no por razones de carácter lógico, sino sólo para conseguir la apariencia de tridimensionalidad.

Realmente, esta forma de actuar es un desperdicio de recursos. El que un objeto tenga que tener apariencia tridimensional no implica necesariamente que tenga que ser definido tridimensionalmente.

La figura 5.17 muestra una imagen de una pared de ladrillo, obtenida mediante un simple **bitmap**[1]. En la figura, puede apreciarse perfectamente la estructura de la pared sin necesidad de recurrir a ningún tipo de técnica 3D.

¿Por qué no utilizar este mismo sistema en el caso tridimensional? Podemos utilizar bitmaps para las superficies rugosas, y proyectarlos sobre los polígonos correspondientes, como muestra la figura 5.18. En cierto modo, es como si estuviéramos "pegando" el bitmap sobre un polígono 3-D.

Todo se trata de una cuestión de escalas. Por ejemplo, un personaje humano dentro del mundo virtual presenta una determinada forma (no excesivamente compleja) visto desde la distancia. Si nos acercamos, podemos empezar a percibir más detalles, como los pliegues de la ropa o el relieve de las facciones. Si nos acercamos aún más, la textura de la ropa y de la piel empiezan a hacerse patentes... Todo ello son características debidas a la tridimensionalidad de los objetos, que podemos definir con la precisión que deseemos.

---

1 Un bitmap es, simplemente, una matriz que indica el color de cada uno de los puntos que componen una cierta imagen. En la imagen de la figura, la pared de ladrillo puede almacenarse en memoria mediante un bitmap que indique el color de cada punto; si en una aplicación se quisieran mostrar paredes de varios tipos distintos, se utilizaría un bitmap diferente para definir cada pared.

Figura 5.17. Ejemplo de textura

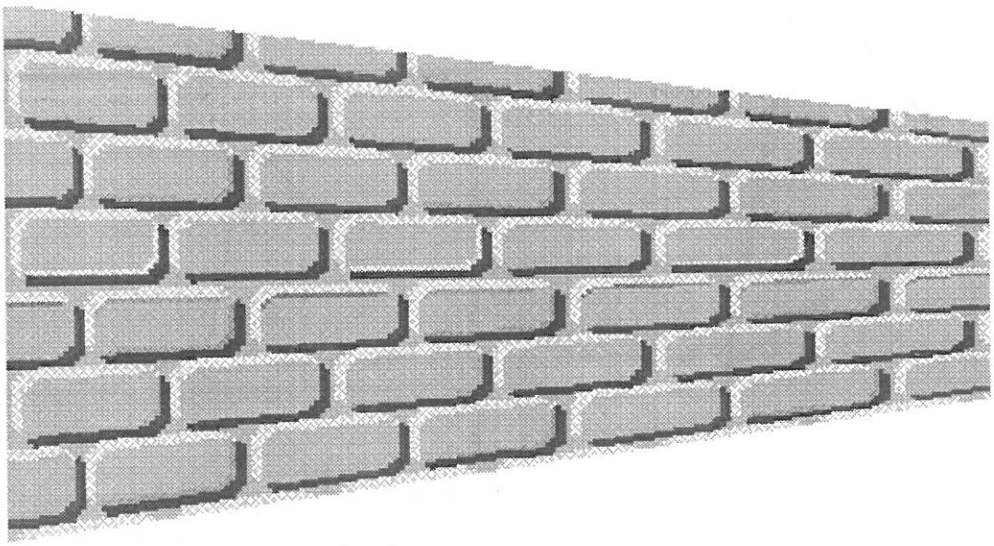

Figura 5.18. Proyección de textura sobre un polígono en perspectiva

¿Cuál es el modo lógico de proceder? Pues utilizar una definición poligonal sólo para la tridimensionalidad de mayor escala, aquélla que realmente tiene que ver con la lógica de la aplicación, y utilizar las técnicas de texturado para simular los aspectos de detalle.

Con las técnicas de texturado se puede mejorar enormemente la calidad de las escenas, porque permiten realzar los aspectos tridimensionales de los objetos sin incrementar las necesidades de procesamiento geométrico. Un ejemplo de hasta qué punto es efectiva esta técnica es el videojuego Ridge Racer, de Namco, un juego de carreras en el que los paisajes presentan un realismo extraordinario gracias a la utilización intensiva de texturados. Asimismo, si el lector ha tenido la oportunidad de jugar a DOOM en su PC, habrá podido comprobar de forma directa las enormes posibilidades que las texturas ofrecen.

¿Cómo se implementa la técnica de texturado? En el caso general, lo que se hace es asignar a cada vértice de un polígono una posición dentro del bitmap de textura correspondiente. Entonces, a la hora de realizar la rasterización del polígono, se calcula (por interpolación) qué pixel del bitmap corresponde a cada pixel del polígono, y se da a éste el color de aquél (ver figura 5.19).

Lógicamente, debido a las operaciones extra de manejo e interpolación del bitmap, la rasterización de los polígonos es más lenta en el caso de las texturas, pudiendo llegar a

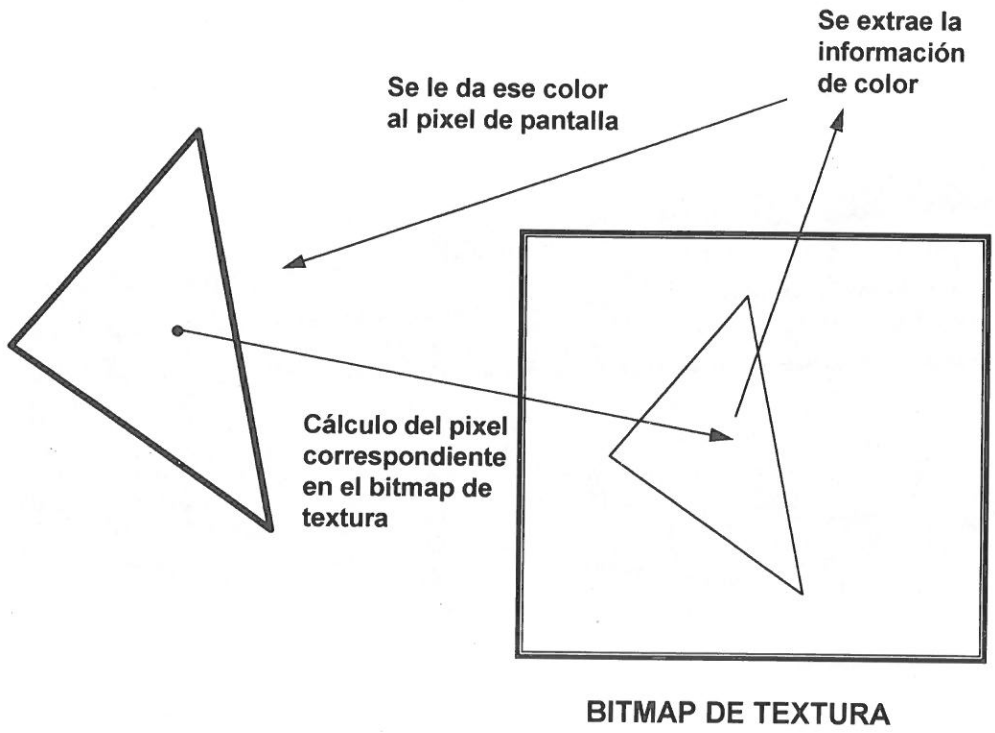

Figura 5.19. Método de aplicación de la textura

consumir un tiempo hasta cinco veces superior (si no se dispone del adecuado soporte hardware). Por eso las texturas encuentran su principal aplicación en el caso de superficies planas y extensas, como la pared de ladrillos, donde lo que hacemos es emplear un único polígono grande, texturado, en lugar de múltiples polígonos pequeños. Pintar un polígono con textura es más costoso que el mismo polígono sin textura, pero pintar un único polígono texturado de gran tamaño es mucho menos costoso que pintar múltiples polígonos pequeños sin textura.

De forma que las texturas tienen sentido principalmente, como ya hemos dicho, en el caso de superficies de gran extensión (paredes, suelos, superficies de mesas) u objetos que, por estar situados a gran distancia, puedan sustituirse por una superficie (por ejemplo, el fondo montañoso de una escena).

En el caso de objetos curvos y complejos (por ejemplo, una figura humana) es necesario calibrar cuidadosamente el uso de texturas. Si con ellas podemos reducir el número de polígonos necesarios para representar el objeto, entonces puede que ahorremos tiempo de síntesis. Si el ahorro en número de polígonos no es grande, entonces las texturas pueden llegar a ralentizar el sistema.

Ahora bien, en este último caso, las texturas siguen teniendo la ventaja de aumentar el realismo de la escena, por ejemplo simulando la piel de un animal o la ropa de una figura humana. Así, es posible que, en ocasiones, merezca la pena introducir texturas aun cuando la síntesis pierda en velocidad.

Además del hecho de que pueden hacer más lento el proceso de síntesis, si no se utilizan con cuidado, las texturas presentan otras varias limitaciones:

- La primera es que, como ya hemos señalado, las texturas sólo imitan la apariencia tridimensional: el polígono sobre el que la textura se proyecta continúa siendo un plano. En el caso de la pared de ladrillos, los ladrillos añadidos mediante una textura no hacen que la pared deje de ser lisa, simplemente hacen que no lo parezca. Como consecuencia, en los sistemas de realidad virtual inmersivos, no es posible percibir mediante la estereoscopía el relieve correspondiente. De forma que, a distancias cortas, el usuario percibiría la pared como lo que en realidad es: como una superficie lisa sobre la que hubieran pegado la fotografía de unos ladrillos.
- Las texturas pueden también hacer perder realismo a las escenas si los juegos de luz no son adecuados. En la figura 5.18 podemos observar que el efecto de las sombras entre los ladrillos corresponde a una situación en que la luz viene de la parte superior izquierda de la página. Cuando más tarde utilicemos la textura sobre una superficie del espacio 3D, dicha superficie podrá o no tener la misma posición relativa con respecto a las fuentes de luz situadas. Si no la tiene, la "sombra" de la textura no se corresponderá con la que el usuario esperaría ver, y la escena puede perder realismo. Normalmente, si limitamos el uso de texturas a grandes superficies estáticas (paredes, fondos montañosos) y la posición de las luces en la escena es fija, se pueden diseñar de antemano las texturas con las sombras adecuadas.
- La tercera limitación se refiere al sombreado. Tal como hemos expuesto el mecanismo de cálculo de la textura de un polígono, el color de cada pixel del polígono vendrá

determinado sólo por la textura, sin que las fuentes de luz intervengan para nada. Como consecuencia, un polígono texturado no presentará variaciones de intensidad de color al moverse en relación a las luces; lo cual también resta realismo a las escenas. Para evitar este efecto, se pueden combinar texturas y sombreados. La forma es la siguiente: para cada pixel del polígono, se calcula cuál es el color a partir del bitmap de textura, y se calcula cuál es el nivel de iluminación del punto interpolando el nivel de iluminación de los vértices (en el caso de sombreado de Gouraud); después se modifica el color del punto de acuerdo con su nivel de iluminación.

- Existe otra limitación, fácilmente resoluble, relativa a la forma de pintar el bitmap en la escena. Imagine, por ejemplo, que tenemos un bitmap cuadrado en el que está representada la figura de un árbol. Dicho árbol se habrá pintado sobre un cierto fondo, por ejemplo blanco. Si pintáramos el bitmap completo tal cual, lo que aparecería en la escena sería un cuadrado blanco con un árbol pintado en medio, es decir, el "fondo" utilizado para el bitmap se pintaría también, tapando los objetos que hubiera detrás.

  Eso no es, obviamente, lo que queremos. Para resolverlo, lo que se hace es elegir un determinado color para el fondo de los bitmpas, que se denomina **color de transparencia**. A la hora de dibujar el bitmap en pantalla, se comprueba cuál es el color de cada punto antes de dibujarlo. Si su color es el de transparencia, entonces es que dicho punto correspondía al fondo del bitmap, por lo que no se dibuja. Esta técnica se utiliza ampliamente en juegos 2D para mover las figuras por la pantalla. Obsérvese que cuando el objeto representado en el bitmap tiene una forma regular y ocupa la totalidad de éste (como en el caso de la pared de ladrillos) no es necesaria esta precaución.
- El último de los problemas es de más difícil solución. Como ya se comentó al hablar de la técnica del z-buffering, no es correcto matemáticamente realizar interpolaciones lineales en los polígonos, una vez que éstos han sido proyectados en pantalla. Como consecuencia, cuando se "pega" un bitmap sobre un polígono, se produce una cierta distorsión. Si los triángulos ocupan pocos pixels en pantalla esa distorsión es pequeña, pero para triángulos grandes puede llegar a hacerse apreciable. En estos casos, es necesario realizar una **corrección de perspectiva**, lo que puede hacerse aplicando fórmulas de interpolación exactas (con el consiguiente incremento en tiempo de computación), o dividiendo esos polígonos de gran tamaño en otros más pequeños, reduciendo así la distorsión. Esta última técnica se conoce con el nombre de **teselación**.

5.5.4. Bitmaps auto-orientables

Existe un caso particular de objetos complejos para los que resulta posible emplear texturas de una forma bastante ingeniosa. Nos estamos refiriendo a la utilización de los llamados bitmaps auto-orientables.

Con esta técnica, los objetos se representan mediante un único polígono con una textura (es decir, mediante un bitmap), el cual se hace rotar alrededor de una posición central a medida que el usuario se mueve, de forma que éste siempre ve el polígono de frente.

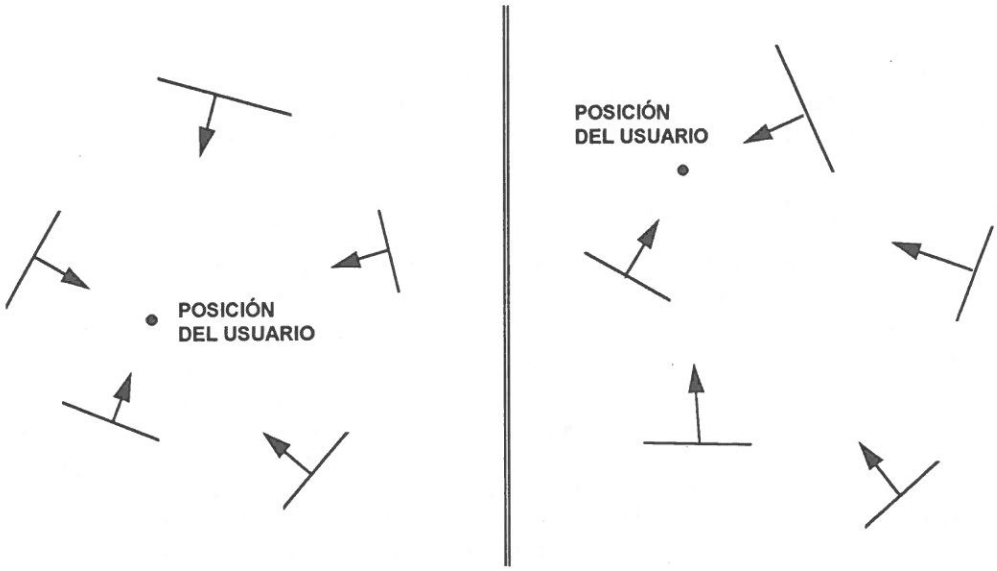

Figura 5.20. Bitmaps auto-orientables

Por ejemplo, cada árbol de una escena puede representarse mediante un único polígono rectangular, que se hace girar sobre un eje situado en el tronco a medida que el usuario se desplaza por la escena. De esta manera, el usuario siempre ve el bitmap del árbol (ver figura 5.20).

Para poder conseguir el efecto de "transparencia" (es decir, que puedan verse a través de las ramas los objetos situados detrás del árbol), se recurre a la técnica de la transparencia, antes mencionada.

La técnica de los bitmaps auto-orientables es especialmente apropiada para los objetos con simetría radial o esférica, por ejemplo árboles, consiguiéndose mejor efecto cuanto más lejos está el objeto. Pero hay programas comerciales (por ejemplo los juegos Wolf 3-D y DOOM, para PC) que han utilizado esta técnica con otro tipo de objetos no simétricos, consiguiendo unos resultados sorprendentes.

5.5.5. Otras técnicas

Dependiendo de la naturaleza de las escenas, pueden existir otras posibles optimizaciones del proceso de síntesis. En aquellos casos, por ejemplo, en que las luces de la escena tienen una posición fija, se puede precomputar la iluminación de los vértices de todos los objetos estáticos, disminuyendo así el tiempo necesario de procesamiento geométrico. Esta técnica se conoce con el nombre de **pre-iluminación**.

Otro conjunto de técnicas ampliamente utilizado es el de las llamadas técnicas de **variación dinámica del nivel de detalle**, que ya mencionamos al principio de este apartado. Dada la amplitud del tema, a estas técnicas se dedica todo el apartado siguiente.

5.6. VARIACIÓN DINÁMICA DEL NIVEL DE DETALLE

Una técnica bastante eficiente para reducir el tiempo de representación gráfica de las escenas consiste en alterar, dinámicamente, el nivel de detalle de los objetos.

Pongamos un ejemplo. Si nuestra escena incluye una serie de casas, aquéllas que estén situadas más lejos del usuario se verán de menor tamaño, es decir, ocuparán en la pantalla menos espacio (menos pixels). En un caso extremo, una casa muy alejada podría ocupar tan sólo una decena de pixels con lo que, por muy perfecta que sea la definición gráfica de la casa, todos los detalles se perderán al realizar la proyección de la escena sobre los pixels que componen la pantalla. En el mundo real sucede exactamente lo mismo: nuestro ojo no es capaz de percibir los detalles de los objetos muy alejados, porque su imagen ocupa una zona demasiado pequeña de la retina, excitando a un reducido número de células.

En este caso, ¿tiene sentido que empleemos una definición gráfica muy precisa de la casa, con miles de polígonos, para al final quedarnos con sólo unos pocos pixels? Puesto que el nivel de detalle se va a perder de todas formas, ¿por qué no ahorrar tiempo y emplear una definición gráfica menos precisa de la casa?

Esto es precisamente lo que hace la técnica de **variación del nivel de detalle**. Para cada objeto, en lugar de disponer de una única definición gráfica, se almacenan varias definiciones, cada una con un nivel de detalle diferente.

Este tipo de técnica puede utilizarse en varias situaciones:

- Variación del nivel de detalle con la distancia.
- Variación con la velocidad.
- Variación con el foco de atención.
- Variación por selección del usuario.
- Variación con la carga de trabajo.

5.6.1. Variación del nivel de detalle con la distancia

Se utiliza un nivel de detalle diferente según los objetos estén cerca o lejos del usuario. Normalmente se emplean dos o, a lo sumo, tres definiciones gráficas de los objetos: una para los situados lo suficientemente cerca del usuario como para que éste sea capaz de percibir los detalles, otra para aquéllos tan alejados que la mayoría de los detalles son indistinguibles y, si acaso, otra para aquéllos objetos situados a una distancia intermedia del observador.

A medida que el usuario se acerca a los objetos, se va pasando de una definición gráfica a otra, para lo cual existen dos métodos distintos:

- **Transición abrupta**, en la cual se conmuta directamente de una a otra definición cuando se alcanza la distancia oportuna (si es preciso, se introduce una cierta

histéresis). Éste es el método más sencillo, pero la sensación que obtiene el usuario es la de que "aparecen" los objetos en pantalla de modo repentino. Aumentando el número de definiciones gráficas utilizadas disminuye este efecto, ya que las diferencias entre ellas son menores.

- **Mezclado**, que se basa en pintar simultáneamente ambas definiciones gráficas (con intensidades diferentes) mientras el usuario se encuentra en una cierta franja de transición. En un extremo de la franja, la imagen es casi idéntica a una de las definiciones gráficas, mientras que en el otro extremo es la otra la que predomina. De esta forma, el usuario percibe una transición más suave, aunque es un mecanismo más costoso que el anterior en cuanto a tiempo de computación.

5.6.2. Variación del nivel de detalle con la velocidad

El nivel de detalle que el ser humano es capaz de percibir es mucho mayor para los objetos estáticos que para los objetos en movimiento. Cuanto mayor es la velocidad de un objeto, menor es el nivel de detalle que podemos captar. En consecuencia, podemos emplear definiciones gráficas diferentes para cuando los objetos se mueven (con respecto al usuario) a distintas velocidades.

Lo mismo cabe decir cuando es el usuario el que se está moviendo. Podemos, por ejemplo, emplear un nivel de detalle bajo mientras que el usuario se desplaza y, una vez que éste se para, volver a pintar la última imagen, esta vez con un nivel de detalle mayor. Esto es lo que se conoce con el nombre de **técnica de refinamientos sucesivos**.

Esta técnica de los refinamientos sucesivos resulta problemática en los sistemas inmersivos basados en visiocasco, porque el usuario no puede mantenerse perfectamente estático y el sistema, como consecuencia, no tiene modo de saber cuándo el usuario se ha parado, para empezar a refinar la escena. La única solución consiste en filtrar de algún modo los movimientos involuntarios de pequeña amplitud que el usuario realiza, pero ese filtrado introduce una latencia adicional en el sistema.

5.6.3. Variación del nivel de detalle con el foco de atención

Esta técnica se utiliza en algunos simuladores de vuelo y se basa en el hecho de que el hombre percibe mayor nivel de detalle de los objetos cuanto más cerca están éstos situados de su eje de visión. Aquellos objetos que se proyectan sobre la zona central de la retina se perciben con un nivel de detalle alto, mientras que los objetos situados en el límite de nuestro campo de visión se perciben mucho más difusamente.

En consecuencia, podemos emplear bandas electro-oculográficas para detectar la posición de los ojos del usuario, con el fin de dibujar con un detalle menor los objetos más alejados del eje de visión, y así reservar una mayor capacidad de proceso para los objetos hacia los que el usuario mira.

5.6.4. Variación del nivel de detalle por selección del usuario

La técnica anterior, de variación del nivel de detalle en función del foco de atención, consistía esencialmente en presentar con un mayor nivel de detalle aquellos objetos de la escena que más interesan al usuario. La utilización de las bandas electro-oculográficas permitía deducir de manera automática qué objeto o conjunto de objetos eran los que el usuario consideraba de mayor interés en un momento determinado.

En las aplicaciones de visualización científica, esta misma técnica puede aplicarse sin necesidad de emplear ningún dispositivo sofisticado como las bandas electro-oculográficas. Podemos hacer que sea el propio usuario quien indique explícitamente qué partes de la imagen le interesan más. Utilizando cualquier dispositivo de control, el usuario puede seleccionar uno o más objetos, y el sistema presentar éstos con un nivel de detalle superior.

5.6.5. Variación del nivel de detalle con la carga de trabajo

Las variaciones en el nivel de detalle también pueden emplearse para adaptar el sistema a la carga total de trabajo existente. Por ejemplo, si el sistema detecta que no le da tiempo a dibujar las escenas, debido a la gran cantidad de objetos existentes o (en los sistemas multiproceso) al poco tiempo de CPU disponible o a cualquier otra causa, una solución consiste en que el sistema pase a usar definiciones gráficas con menor nivel de detalle para todos los objetos o para un subconjunto de los mismos. Después, una vez que vuelva a haber capacidad de procesamiento suficiente, el sistema puede volver a emplear las definiciones más completas.

Este método resulta muy útil en aquellas aplicaciones donde no se puede prever a priori la carga de trabajo máxima del sistema. Sin embargo, tiene la desventaja de que el usuario percibe variaciones en la calidad global de la imagen, lo que puede ser inaceptable en otras aplicaciones, por ejemplo los juegos comerciales. Este otro tipo de aplicaciones, por ello, se desarrollan de manera que la carga máxima de trabajo permanezca siempre bajo condiciones controladas.

5.7. EL PROCESO DE SÍNTESIS (II)

5.7.1. El proceso de síntesis revisitado

Como puede verse, el proceso de síntesis puede llegar a ser todo lo complejo que se quiera, dependiendo de la funcionalidad que deseemos, del soporte hardware del que dispongamos y de lo óptimo que queramos que el proceso sea.

La figura 5.21 muestra un resumen del algoritmo, en el que se han incluido (sobre el proceso básico), las optimizaciones más comúnmente empleadas en los sistemas de RV actuales: selección, eliminación de caras ocultas, texturado y variación del nivel de detalle con la distancia.

```
borrar el buffer donde vayamos a dibujar la nueva imagen;
inicializar el z-buffer;
para cada objeto de la escena{
  calcular la matriz M de posición con respecto al observador;
  si el volumen de recubrimiento no corta pirámide de visión
    -> DESECHAR EL OBJETO Y PASAR AL SIGUIENTE
  si distancia al observador mayor que LIMITE{
    seleccionar representación gráfica con poco nivel de detalle;
  }
  en caso contrario{
    seleccionar representación gráfica con alto nivel de detalle;
  }
  para cada polígono del objeto{
    calcular, mediante M, la posición con respecto al observador;
    si el polígono es una cara oculta
      -> DESECHAR EL POLIGONO Y PASAR AL SIGUIENTE
    si el polígono cae fuera del ángulo de visión
      -> DESECHAR EL POLIGONO Y PASAR AL SIGUIENTE
    si cae parcialmente fuera{
      recortar la parte sobrante;
    }
    para cada vértice del polígono{
      calcular su grado de iluminación;
      proyectar el vértice sobre la pantallal;
    }
    para cada pixel del polígono proyectado{
      calcular su Z, por interpolación de la Z de los vértices;
      si es menor que la almacenada en el z-buffer{
        sustituir el valor del z-buffer por Z;
        calcular su iluminación, interpolando la de los vértices;
        determinar color según la textura;
        modificar color según el grado de iluminación;
        dar al pixel dicho color;
      }
    }
  }
}
presentar la nueva imagen;
```

Figura 5.21. El proceso de síntesis al completo

5.7.2. Problemas del proceso de síntesis

El proceso de la síntesis de imágenes no es un proceso exacto. Como habrá podido ver por nuestras explicaciones, el proceso está plagado de fuentes de inexactitud.

Para empezar, el proceso parte de una definición imperfecta de los objetos, mediante un conjunto poligonal. Cuantos más polígonos se utilicen, más perfecta puede ser esta aproximación, pero más tiempo consumirá el algoritmo en labores de procesamiento geométrico. Dado que la capacidad de proceso de cualquier sistema es limitada, estamos forzados a aceptar un cierto grado de imprecisión.

La segunda fuente de imprecisión proviene de las inexactitudes en los cálculos intermedios. En la fase de procesamiento geométrico, todos los cálculos realizados lo son en punto flotante, y la precisión de los procesadores es limitada. Eso origina errores de acarreo que se arrastran de un paso del proceso al siguiente. En la fase de rasterización se emplea aritmética de números enteros, y se acumulan nuevos errores debido a que los resultados de las interpolaciones (que incluyen operaciones de división) tienen que ser forzosamente redondeados.

Finalmente, una tercera fuente de imprecisiones nace del hecho de que la salida utilizada (es decir, la pantalla) también es imperfecta. Las pantallas poseen una resolución limitada: a la hora de dibujar una imagen, lo que en realidad estamos dibujando es un conjunto de puntos aislados de la misma. Es decir, estamos efectuando un muestreo de la señal, en el cual se pierde información.

En esta sección vamos a ver algunas de las técnicas empleadas en los sistemas gráficos para tratar con este tipo de errores.

5.7.3. Anti-aliasing

La principal consecuencia de la limitada resolución de las pantallas es que se introduce una distorsión en los contornos de las imágenes al ser éstas dibujadas. Esta distorsión se conoce con el nombre de aliasing.

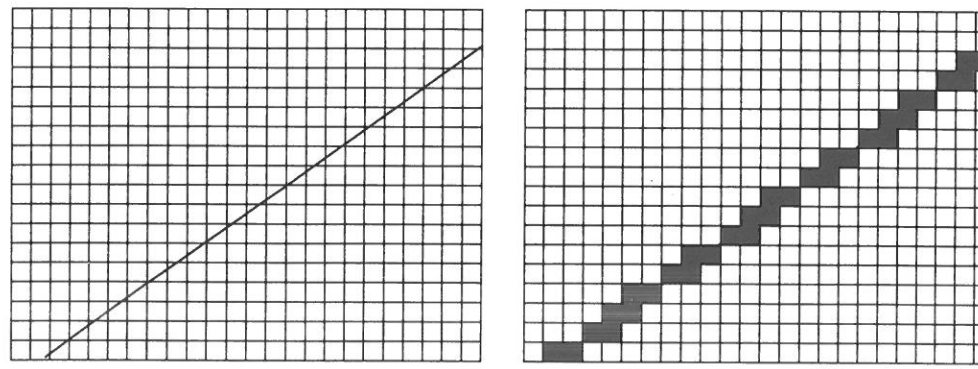

Figura 5.22. Efecto del aliasing sobre una linea

Puede comprobarse fácilmente el efecto del aliasing sin más que dibujar una línea recta. La resolución limitada hace que aparezca un "efecto escalera" (figura 5.22) tanto más acusado cuanto más baja sea la resolución de la pantalla. El efecto se puede apreciar mejor para las líneas de ángulo muy próximo a la vertical o a la horizontal.

Este efecto escalera es uno de los más llamativos, pero no el único. El aliasing también provoca, por ejemplo, que aparezcan distorsiones cuando múltiples líneas convergen en la distancia.

Para las imágenes estáticas, en los sistemas de baja resolución (como la mayoría de sistemas basados en visiocasco), la apariencia de los objetos puede dejar bastante que desear, debido al fenómeno del aliasing. Sin embargo, si la resolución es media o alta (como pueda ser un sistema de sobremesa con resolución VGA o superior), el efecto del aliasing carece apenas de importancia. Así pues, en el caso de imágenes estáticas basta con aumentar la resolución a unos niveles normales para corregir el problema.

Sin embargo, cuando lo que se nos presenta en pantalla no es una imagen estática sino una sucesión de ellas, la cosa cambia. Porque al problema de la distorsión debida al aliasing se añade el hecho de que esa distorsión no es igual entre una imagen y la siguiente (debido a los errores de redondeo), y el resultado es que (incluso para pantallas de alta resolución) aparecen puntos parpadeantes en la escena allí donde dos líneas se cortan, y figuras de distorsión danzantes allí donde las líneas convergen en la distancia. Estas distorsiones "móviles" son lo que más daña la apariencia del conjunto.

En consecuencia, si se desea una imagen de alta calidad, es necesario introducir en nuestro proceso de síntesis algún mecanismo que corrija la distorsión.

Existen dos maneras de hacerlo. El resultado de ambas es muy similar, y la principal diferencia radica en la manera de implementarlas:

5.7.3.1. Sobremuestreo

Ésta es la técnica más utilizada. Consiste en muestrear las imágenes con una resolución superior a la que la pantalla permite. De esta manera, a cada pixel de pantalla le corresponde no una muestra, sino un grupo de ellas. Después de calcular el color de cada muestra, se asigna el color al pixel de la pantalla "ponderando" las muestras correspondientes.

La figura 5.23 muestra esta técnica para un factor de sobremuestreo de 2 (es decir, a cada pixel se le hace corresponder un conjunto de 2x2 muestras). En la figura, se asigna al pixel el color negro sólo si al menos dos de las muestras son de dicho color. Si ninguna muestra es negra, se da al pixel el color blanco. Si hay una sola muestra, el pixel toma un color intermedio.

5.7.3.2. Filtrado

La segunda de las técnicas se basa en efectuar un post procesamiento de la imagen, una vez que ésta ha sido completamente generada. Ese proceso de filtrado consiste en una serie de cálculos matemáticos que difuminan los contornos de los objetos, lo que tiene como

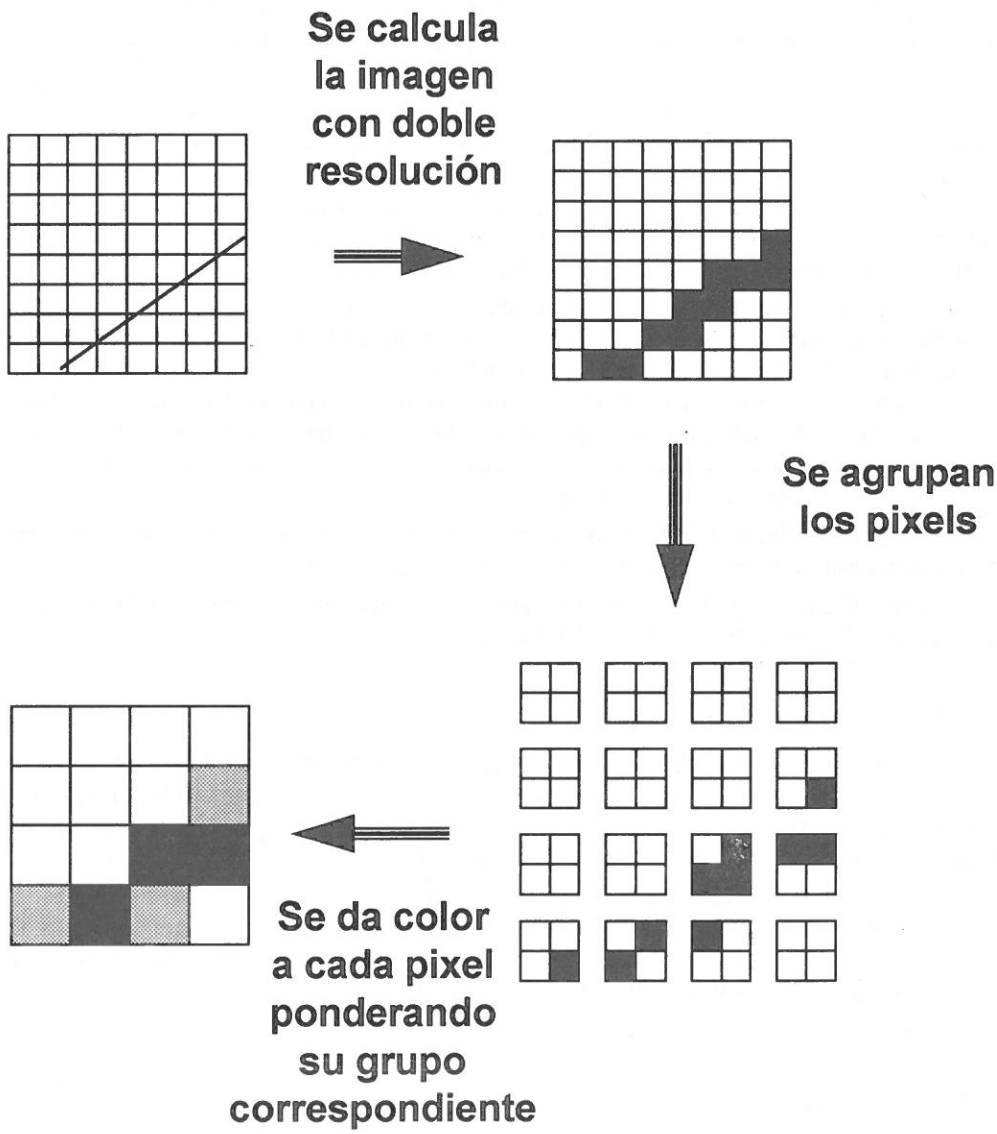

Figura 5.23. Antialiasing por sobremuestreo

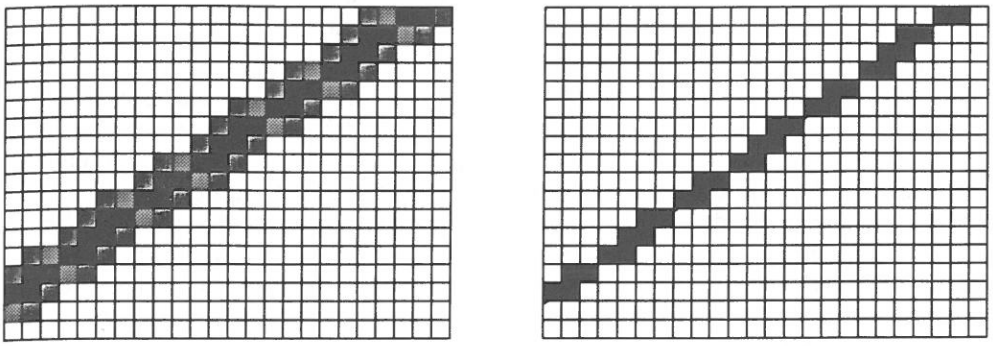

Figura 5.24. Filtrado de la imagen

consecuencia que disminuya el efecto del aliasing. La figura 5.24 muestra un ejemplo de filtrado de la señal. Como puede verse, el filtrado no hace más que asegurar que las transiciones de un color a otro no sean excesivamente bruscas.

Resulta evidente que cualquier medida que tomemos para amortiguar el efecto del aliasing redundará en un incremento en el tiempo de rasterización. Un sobremuestreo de factor 2, por ejemplo, implicará multiplicar por cuatro el tiempo de rasterización. Por eso la mayoría de aplicaciones de RV no profesionales no suelen incluir mecanismos de anti-aliasing, excepto en el caso de las texturas, como a continuación se indica.

5.7.3.3. El problema del aliasing en las texturas

El aliasing no sólo afecta a los contornos de los objetos. También cuando se emplea una textura, el resultado que finalmente aparece en la pantalla es el resultado de un muestreo del bitmap original, y ese muestreo hace que aparezca el fenómeno del aliasing, estropeando la apariencia gráfica de la textura. Para resolver el problema, se puede utilizar cualquiera de las dos técnicas mencionadas: sobremuestreo y post-filtrado.

Existe, sin embargo, una tercera técnica, empleada en juegos como DOOM, que consiste en efectuar un **pre-filtrado** de la textura, en el momento de diseñarla. El pre-filtrado garantiza que la textura no contendrá variaciones bruscas de color, lo que reducirá el efecto del aliasing cuando luego esa textura sea proyectada.

Si se quiere aumentar aún más la calidad, puede recurrirse a la utilización de **mipmaps**, que es una variante del proceso de sobremuestreo. Recordemos que una de las técnicas empleadas para reducir la complejidad de una escena era la variación del nivel de detalle con la distancia. Con dicha técnica, el sistema mantenía varias versiones de un mismo objeto y, a la hora de sintetizar la escena, elegía entre una y otra versión en función de lo alejado

que el objeto estuviera. En el caso de las texturas, puede hacerse algo similar, pero no para reducir la complejidad, sino con el objeto de reducir el aliasing.

Imaginemos que la textura original es de 64x64 bits. En ese caso, podemos almacenar (antes de comenzar la ejecución) varias versiones reducidas de la textura, además de la original: una versión de 32x32, otra de 16x16, otra de 8x8, etc... Cada una de dichas versiones estará calculada de forma que cada pixel de la versión de 32x32 se obtiene ponderando los 2x2 pixeles correspondientes en la versión de 64x64; cada pixel de la versión de 16x16 se obtiene ponderando los 2x2 pixeles correspondientes en la versión de 32x32, etc... De esta forma, a la hora de proyectar la textura, elegiremos una u otra resolución en función de la superficie que la proyección ocupe en pantalla. Se trata, como vemos, de un sobremuestreo, pero con la diferencia de que está ya precalculado, ahorrándonos, por tanto, tiempo de ejecución.

5.7.4. ¿Cómo definir un objeto de manera perfecta?

El hecho de definir los objetos mediante conjuntos de polígonos tiene como consecuencia que los contornos de los objetos (incluso de aquellos cuyas superficies son curvas) aparezcan plagados de aristas. Esto se debe a que el definir el objeto mediante polígonos equivale, de hecho, a una operación de muestreo. En lugar de tomar todos los puntos de la superficie del objeto, lo que hacemos es quedarnos con una serie limitada de puntos, que son los vértices de los polígonos. Y ese proceso de muestreo hace que se pierda información.

¿Existe alguna manera de evitar esto? Pues está claro que sí: si no muestreamos los objetos, no se producirá la pérdida de información consiguiente.

En lugar de emplear una aproximación poligonal, podemos utilizar una **definición matemática** del objeto, mediante una ecuación o un conjunto de ellas. Una esfera, por ejemplo, podríamos representarla por su correspondiente ecuación de tres variables, X, Y y Z:

$$Radio^2 = X^2 + Y^2 + Z^2$$

Este sistema presenta dos inconvenientes y dos ventajas. El primer inconveniente es que, para cualquier objeto que no sea excesivamente simple, la definición matemática de las formas correspondientes puede ser tremendamente compleja. De hecho, para casi todos los objetos del mundo real, lo más que puede conseguirse es una buena aproximación, pero no una definición matemática perfecta.

El segundo inconveniente concierne a la implementación del proceso de síntesis. Si partimos de una definición matemática de los objetos, efectuar una proyección de los mismos sobre la pantalla puede ser inabordable (para un caso general) por la complejidad de los cálculos matemáticos implicados. Como consecuencia, lo que se hace es invertir el proceso y recurrir a la técnica conocida como **trazado de haces** (ray-tracing). Básicamente, esta técnica consiste en lo siguiente:

- Para cada pixel de la pantalla, calculamos el haz o línea que, partiendo del ojo del observador, pasa por el pixel.

- Miramos a ver a qué objeto intersecta dicho haz y en qué punto lo hace (a partir de la definición matemática del objeto).
- Una vez obtenido el punto, calculamos el color con que se debe mostrar dicho punto en la pantalla y damos dicho color al pixel.

Obsérvese que, en lugar de ir de los objetos a la pantalla (proyectando aquéllos), lo que se hace es ir en sentido inverso, de la pantalla hacia los objetos. Esta técnica se utiliza ampliamente en los programas de animación profesionales, porque partiendo de una definición matemática de los objetos se puede obtener una excelente calidad de imagen.

Sin embargo, esta forma de proceder es inviable en aplicaciones de tiempo real, como son las de RV, debido al elevado número de cálculos matemáticos requeridos. No se trata sólo de calcular la intersección de un rayo con una superficie matemática (lo cual puede ser de por sí muy complejo) sino que, a priori, no sabemos con cuál de los múltiples objetos de una escena intersectará el rayo en primer lugar, lo que nos fuerza a determinar no una sino múltiples intersecciones. Para más desgracia, este método no se presta a ser implementado en hardware, al contrario de lo que sucede con las interpolaciones poligonales. El resultado es que, cuando se sintetiza una escena compleja mediante la técnica de trazado de haces, no es infrecuente que más de las tres cuartas partes del tiempo de proceso se dediquen al cálculo de intersecciones.

En consecuencia, no es posible solventar el problema para aplicaciones de tiempo real como las que nos ocupan. El uso de conjuntos poligonales en los sistemas de RV es obligado. Sin embargo, el sistema de definición matemática tiene su importancia en los sistemas de RV, por lo que a continuación vamos a ver.

Para empezar, el método de definición matemática permite ahorrar memoria. Almacenar la definición matemática de la esfera requiere un espacio de memoria despreciable, mientras que almacenar las coordenadas de todos los vértices de la misma esfera muestreada puede ocupar decenas de Kbytes.

Pero, además, la definición matemática resulta mucho más apropiada en aquellos casos donde lo que se requiere es editar un objeto, y no simplemente dibujarlo. Supongamos, por ejemplo, que queremos modificar el tamaño de una determinada zona, casi esférica, de un objeto. Si el objeto está definido de manera poligonal, tendremos que retocar manualmente las posiciones de todos y cada uno de los vértices situados sobre la esfera. Si disponemos, por el contrario, de una definición matemática del objeto, la modificación puede ser tan simple como cambiar el radio de la esfera mediante un comando.

Con los dibujos en dos dimensiones sucede exactamente lo mismo. Cambiar el tamaño de un cuadrado con el programa CorelDraw es sencillísimo: basta con arrastrar con el ratón hasta obtener la longitud de lado requerida, encargándose el programa de dibujar automáticamente el cuadrado nuevo. Ello sucede porque lo que CorelDraw almacena es la propia definición del cuadrado (su posición y el tamaño del lado), y no sus puntos componentes.

Sin embargo, con PaintBrush (o cualquier otro editor de bitmaps) tenemos que cambiar uno por uno cada uno de los pixels, porque el cuadrado está almacenado como una simple sucesión de puntos, **habiéndose perdido la información de la estructura**. Para PaintBrush,

cada punto no guarda relación con los otros, por lo que no puede ayudarnos a automatizar las tareas de modificación.

Por eso los programas de modelado tridimensional profesionales utilizan, siempre que es posible, una forma u otra de definición matemática.

En consecuencia, durante la fase de desarrollo de la aplicación, lo mejor es utilizar programas de modelado que permitan tantas posibilidades de edición como sea posible, lo cual implica que se utilicen definiciones matemáticas de los objetos. Ahora, una vez desarrollada la aplicación y creados los gráficos correspondientes, las figuras así creadas se muestrean y se almacenan en forma de conjuntos poligonales, que son los que el proceso de síntesis de las imágenes utilizará.

6
Estación de proceso

Habiendo completado el repaso de cómo funciona el proceso de la síntesis de imágenes, podemos comenzar nuestro análisis de las estaciones de proceso utilizadas en los sistemas de RV.

Comenzaremos echando un vistazo a los cometidos que la estación de proceso tiene asignados, y viendo cuál es el funcionamiento típico de la estación durante la ejecución de una aplicación de RV. Expondremos después la arquitectura hardware básica de una estación de proceso, y veremos cuáles son (y qué características tienen) las estaciones de proceso comerciales más utilizadas, prestando especial atención a las estaciones basadas en PC. Finalmente, volveremos a analizar la arquitectura de una estación de proceso típica, pero esta vez desde el punto de vista software, exponiendo en detalle lo referente a cada uno de los módulos software que intervienen en la ejecución de las aplicaciones de RV.

6.1. FUNCIONES

La estación de proceso es el corazón del sistema de realidad virtual. A ella corresponde controlar los distintos elementos que componen el sistema y realizar los cálculos necesarios para gestionar el mundo virtual. Tres son las funciones primordiales que realiza.

La primera de ellas se refiere al mantenimiento del mundo virtual. En cierto sentido, la estación de proceso no es más que un simulador: en su memoria guarda el estado de un mundo virtual formado por una serie de objetos, cada uno de los cuales posee un cierto comportamiento. La estación de proceso se limita a ir modificando de manera continua el estado del mundo virtual, de acuerdo con el comportamiento de los objetos que lo componen. Esta es la **función de simulación**.

El comportamiento de los objetos vendrá determinado por el estado del mundo virtual y por los datos que se reciban como entrada, que pueden ser medidas u órdenes dadas por el usuario. En consecuencia, la estación de proceso debe realizar una serie de labores de **control de entrada**, para recibir los datos pertinentes y así poder determinar el comportamiento de cada objeto.

Finalmente, el sistema debe entregar al usuario la información (visual, sonora, táctil o de otro tipo) que éste precise para conocer el estado del mundo virtual. El usuario utilizará esta información para tomar decisiones en cuanto a lo que quiere hacer. La estación de proceso debe, en consecuencia, realizar una serie de labores de **control de salida**, para poder

entregar al usuario dicha información. Entre estas labores destaca, por su importancia, la **síntesis de las imágenes**, por cuanto que el usuario recibe la mayor parte de la información haciendo uso del sentido de la vista.

Simplificando mucho, podríamos decir que la estación de proceso ejecuta de manera perpetua un bucle formado por los pasos siguientes:

ENTRADA	⇒	• leer las órdenes del usuario; • realizar las medidas del mundo exterior que sean precisas.
LOCALIZACIÓN	⇒	• determinar la nueva posición del usuario.
SIMULACIÓN	⇒	• determinar las acciones de cada uno de los objetos del mundo virtual de acuerdo con el estado del sistema, con las órdenes del usuario y con el propio comportamiento de los objetos; • determinar las interacciones producidas entre los objetos; • calcular los cambios en la posición y apariencia gráfica de cada objeto, resultantes de las acciones e interacciones que hayan tenido lugar.
SÍNTESIS DE IMAGEN	⇒	• mostrar al usuario la nueva situación de la escena, generando los gráficos correspondientes.
SALIDA	⇒	• controlar los dispositivos de generación de sonido, realimentación táctil, elementos mecánicos y otros, de acuerdo con el nuevo estado del mundo virtual.

En cierto sentido, la tarea de localización forma parte de la entrada, al igual que la tarea de representación forma parte de la salida, pero he preferido separar las dos etapas del resto de los controles de entrada y salida por la trascendencia que ambas tienen.

En los capítulos anteriores hemos ya analizado los dispositivos de entrada y salida existentes, las técnicas de localización hoy en día empleadas y los pasos del proceso de síntesis de la imagen. Para completar el panorama, nos queda por consiguiente analizar la parte central de ese bucle que la unidad de proceso ejecuta: la etapa de simulación.

6.1.1. La tarea de simulación

La tarea de simulación incluye tres diferentes actividades:

- Simulación del comportamiento autónomo de los objetos.
- Gestión de las interacciones entre objetos.
- Gestión de las variables globales de la escena.

Con respecto al comportamiento, cada objeto del sistema reaccionará de una cierta manera a los eventos que tengan lugar a su alrededor en el mundo virtual. Dependiendo del lenguaje de programación, del sistema de desarrollo utilizado y del propio enfoque que el programador haya adoptado, ese comportamiento podrá estar definido de una manera más o menos formal y más o menos explícita. Pero, en cualquier caso, el programa de RV contendrá siempre las instrucciones necesarias para indicar al sistema qué hacer con cada objeto ante cada circunstancia.

El comportamiento del objeto dependerá de tres factores fundamentales: su estado interno, el estado del mundo virtual circundante y las órdenes que el usuario emita. Todos esos factores se combinan de acuerdo a un conjunto de reglas de comportamiento predefinidas y, como resultado, el objeto ejecutará una serie de acciones.

En un simulador de vuelo, por ejemplo, el comportamiento de la aeronave puede depender del estado de la propia aeronave (por ejemplo, si el piloto automático está o no conectado), de ciertas variables del entorno (como la velocidad del viento) y de las acciones que el usuario realice (como mover el volante de conducción). En este caso concreto, las reglas de comportamiento incluirán una compleja simulación de las leyes mecánicas que regulan el movimiento de una aeronave y darán como resultado un cierto desplazamiento del avión.

Las acciones que un objeto puede realizar, como resultado de la aplicación de las reglas de comportamiento, pueden clasificarse en cuatro diferentes categorías:

- **Acciones internas**: cambios en el estado interno del objeto que no son observables. Esos cambios implican que el estado interno del objeto se modifica, lo que normalmente influirá en su comportamiento posterior.

  Ejemplo: como resultado de una acción del usuario, el simulador de vuelo puede poner a cero la variable que indica que el piloto automático está activo.

  La implementación de las acciones internas consiste en un simple cambio de los valores de las variables correspondientes al objeto. Si estamos trabajando en algún lenguaje orientado a objetos, como C++, una acción interna equivale a un cambio en los valores de los atributos del objeto.
- **Acciones observables**: cambios en los atributos "observables" del objeto, como la representación gráfica o sonora.

  Ejemplo: el simulador de vuelo puede emitir un aviso acústico para indicar una alarma en la aeronave, o puede apagar el aviso luminoso del piloto automático cuando el usuario lo desconecta.

  La implementación de las acciones observables puede conllevar cambios en las variables del objeto, pero además se tendrán que efectuar las llamadas a las correspondientes librerías de los dispositivos de sonido, sistema gráfico y otros dispositivos de salida. Esas llamadas pueden ser directas (se efectúa la llamada como parte de la función de comportamiento del objeto) o indirectas (la función de comportamiento se limita a cambiar una variable que indica, por ejemplo, cuál es la representación gráfica del objeto y, posteriormente, el proceso de síntesis consulta

esa variable para sintetizar la escena). En el apartado siguiente analizaremos más en detalle este tipo de acciones, al hablar del tema de la animación.

- **Acciones dependientes del contexto**: cambios en aquellos atributos del objeto que están sujetos a posibles interacciones con otros objetos.

  Ejemplo: el cambio de posición de un objeto, que implica la posibilidad de colisión con otros objetos del mundo virtual.

- **Acciones externas**: acciones que un objeto ejecuta sobre otros.

  Ejemplo: el objeto realiza un disparo, o envía un mensaje a otro.

Tanto las acciones internas como las observables son acciones no sujetas a ningún tipo de restricción (más que las marcadas por la propia función de comportamiento del objeto). Sin embargo, las acciones dependientes del contexto y las externas son acciones que pueden verse influidas por factores ajenos al objeto.

Por ello, además de simular el comportamiento de cada objeto del mundo virtual por separado, es necesario simular las posibles interacciones entre unos objetos y otros. Tomemos como ejemplo la posición de un objeto dentro de la escena: no puede variar arbitrariamente el valor de su atributo de posición. Al intentarse situar en una posición determinada del mundo virtual, es posible que se encuentre con que esa posición ya está ocupada por otro objeto. Puede suceder que en un cierto mundo virtual esté permitido que dos objetos se solapen, pero en la mayoría de las aplicaciones (al igual que sucede en el mundo real) el principio de la impenetrabilidad de los cuerpos suele ser inviolable.

Si dejáramos a los objetos actualizar libremente su variable de posición, podríamos encontrarnos fácilmente con violaciones de la lógica del mundo virtual. En consecuencia, la aplicación de RV debe incluir los mecanismos necesarios para asegurarse de que esto no suceda. Por ejemplo, comprobando la validez de cualquier cambio de posición de un objeto antes de llevarlo a cabo.

Por eso decimos que un cambio de posición es una acción que depende del contexto. La aplicación de RV deberá proporcionar todos los mecanismos necesarios para regular este tipo de interacciones entre los objetos.

Por lo que se refiere a las acciones externas, es decir, a las acciones que un objeto ejecuta sobre otro, la situación es parecida. Si un objeto dispara a otro, por poner un ejemplo, el efecto de la acción va a depender de una serie de factores externos al objeto que disparó: por ejemplo, que haya obstáculos que impidan que el disparo llegue a su destino. Por consiguiente, el sistema deberá, también en este caso, efectuar una serie de comprobaciones sobre la validez o el éxito de las acciones.

Además de realizar la simulación del comportamiento e interacciones entre los objetos, lo normal es que la aplicación de RV deba gestionar determinadas variables de carácter global. Piénsese, por ejemplo, en la variable que indica el marcador en un juego. En cada ciclo de simulación, la aplicación deberá comprobar las condiciones que pueden dar a lugar a que el valor de dicha variable se modifique.

6.1.2. Animación

Uno de los aspectos más relevantes dentro de la simulación del comportamiento de un objeto es el que se refiere a los cambios en los atributos observables del objeto, y, especialmente, en lo que se refiere a la representación gráfica del mismo.

En el capítulo anterior, al hablar del proceso de síntesis, hemos explicado en detalle cuáles son los pasos que un sistema de RV sigue para sintetizar **una** imagen, correspondiente al estado que el mundo virtual tiene en **un** instante determinado. Pero como el estado de un mundo virtual va evolucionando a lo largo del tiempo, lo que cualquier aplicación de RV va a generar es una sucesión de imágenes diferentes. Hora es ya de que expliquemos cómo.

Hemos visto que, en cada ciclo del bucle, el sistema realiza la simulación del mundo virtual. Como resultado, los objetos pueden variar sus posiciones dentro del mundo virtual, pueden cambiar de aspecto gráfico, etc... Una vez terminada la simulación y determinados todos los cambios que han tenido lugar, comienza el proceso de síntesis de la escena. Acabada la síntesis, se vuelve a realizar la simulación y tienen lugar cambios adicionales. Después se sintetiza la nueva escena, y así sucesivamente.

6.1.2.1. Objetos animados e inanimados

Existen objetos cuya forma (o, más generalmente, su definición gráfica) no sufre modificaciones durante la ejecución de la aplicación. Es el caso, por ejemplo, de una escena que incluya una serie de farolas: lo normal es que la farola sea un mero elemento decorativo que ni interviene en la escena ni se ve afectado por las acciones de otros objetos, por lo que, pase lo que pase, la farola va a tener siempre el mismo aspecto.

Esto es lo que denominamos **objeto inanimado**: aquél cuya definición gráfica permanece constante mientras que la aplicación se está ejecutando. Es importante observar que esto no guarda relación con el hecho de que un objeto pueda, o no, moverse. Puede haber objetos inanimados que, sin embargo, se muevan (como por ejemplo una mesa que puede ser arrastrada por la habitación). Del mismo modo, hay objetos inmóviles que no tienen por qué ser inanimados (por ejemplo, un árbol cuya definición gráfica va variando para representar el efecto del viento en las ramas, aunque el árbol, como un todo, permanece siempre en la misma posición).

Existen otros objetos, por contraposición a los objetos inanimados, cuya definición gráfica no tiene por qué ser constante. En general, casi todos los objetos que intervienen de un modo u otro en el aspecto "interactivo" de la aplicación (el héroe y los monstruos del juego, los objetos representativos de hombres, animales o vehículos, etc...) suelen sufrir variaciones en su definición gráfica, para reflejar sus movimientos o sus acciones. También, determinados elementos "decorativos" (el árbol cuyas ramas se mecen con el viento, la nube cuya forma cambia, el fuego que arde en una chimenea, etc...) pueden cambiar de definición gráfica, simplemente con el fin de mejorar el aspecto estético de la aplicación.

Cuando la definición gráfica de un objeto no es constante, decimos que se trata de un **objeto animado**.

Puesto que los objetos animados pueden cambiar de representación gráfica, es necesario que el sistema determine cuál es la definición gráfica que corresponde a cada objeto, en cada momento.

Al igual que sucedía con los métodos de definición gráfica de los objetos, el sistema de realidad virtual tiene dos maneras de determinar los cambios que dicha definición sufre durante la ejecución de la aplicación.

Si la forma de nuestros objetos está definida de manera matemática, nada nos impide que empleemos el mismo método con los cambios de dicha forma. Podemos especificar, por ejemplo, que cada vez que se dan determinadas condiciones se realicen ciertos cambios en las ecuaciones que definen al objeto.

Lo normal, no obstante, en las aplicaciones de tiempo real es que los objetos estén definidos mediante un conjunto de polígonos. En este caso, los cambios en la definición gráfica se especifican mediante un **diagrama de animación**.

Un diagrama de animación indica al sistema cuál es la definición gráfica que hay que usar en cada momento, y estará compuesto por una o varias **secuencias de animación**.

La figura 6.1 muestra un ejemplo de secuencia de animación bidimensional. La secuencia muestra un conjunto de "instantáneas" correspondientes al desplazamiento de una figura. Cada vez que hay que dibujar de nuevo la escena, el sistema sustituye la definición gráfica de la figura por la siguiente de la secuencia, lo que permite conseguir la sensación de movimiento. En el caso tridimensional, las secuencias de animación funcionan de la misma manera: el sistema va sustituyendo una figura por otra, es decir, una definición poligonal del objeto por otra definición.

El diagrama de animación de un objeto estará compuesto, como ya hemos dicho, de una o más secuencias de animación, entre las cuales irá circulando el objeto de acuerdo con lo que su función de comportamiento marque (ver figura 6.2).

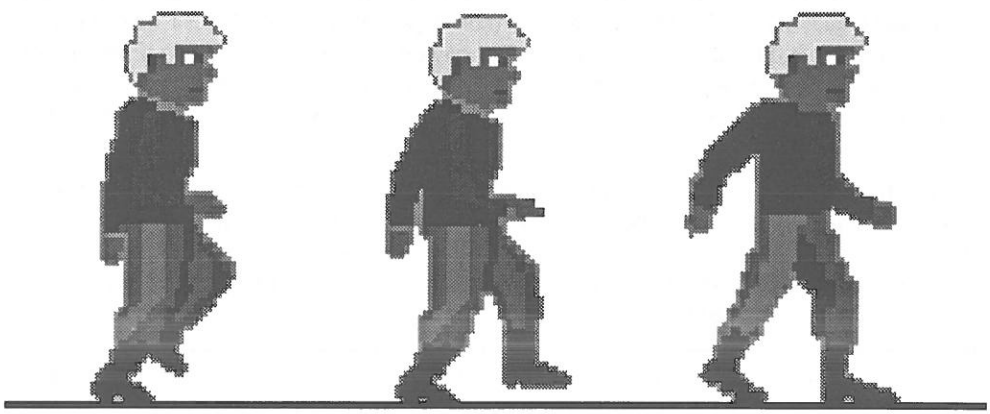

Figura 6.1. Secuencia de animación bidimensional

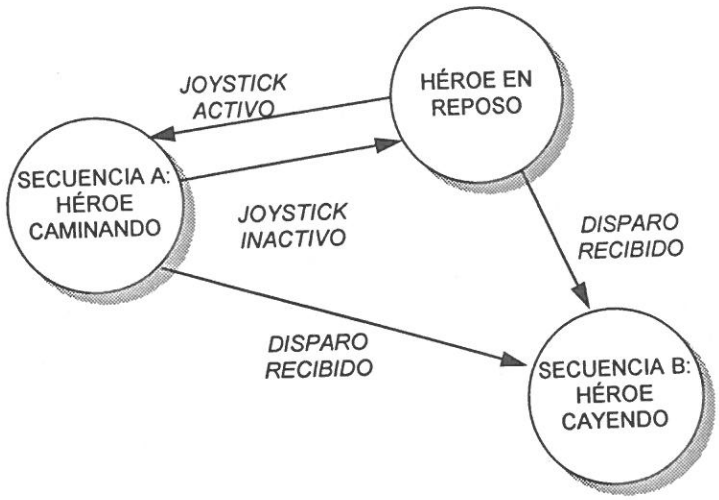

Figura 6.2. Diagrama de animación

6.1.3. Objetos simples y compuestos

Uno de los problemas que presenta la utilización de secuencias de animación son los grandes requerimientos de memoria asociados. En los videojuegos 2D comerciales, no es infrecuente que cada uno de los personajes principales tengan más de un centenar de representaciones gráficas distintas, correspondientes a las distintas acciones que pueden realizar.

En una aplicación 3D, el número de representaciones diferentes de un personaje puede ser hasta tres veces menor. La razón estriba en que en una animación 2D, es necesario prever secuencias de animación distintas para las distintas direcciones de movimiento (horizontal, arriba y abajo); mientras que en una aplicación 3D la dirección del movimiento es indiferente (porque el algoritmo de síntesis ya se encarga de proyectar de la manera adecuada), por lo que con una única secuencia basta para cualquier dirección.

Sin embargo, aun cuando el número de representaciones gráficas de un objeto sea menor, cada una de ellas ocupa mucha más memoria. A título de ejemplo, con 2 Kbytes se pueden definir bitmaps de calidad más que aceptable para un personaje de un juego 2D para PC, mientras que no es raro que la definición poligonal de un personaje típico 3D, no excesivamente complejo, pueda requerir por encima de los 200 polígonos, cada uno de ellos ocupando 100 bytes.

En consecuencia, se suelen emplear diferentes técnicas para optimizar la ocupación de memoria. La más obvia es la de evitar la duplicación de información: si en una misma escena aparecen varios objetos del mismo tipo (por ejemplo varios monstruos iguales), solamente se mantendrá un diagrama de animación para todos ellos, y no un diagrama por cada entidad.

También se pueden, por ejemplo, descomponer determinados objetos en sus partes constituyentes. Así, un objeto que represente a un vehículo podrá descomponerse (a efectos

gráficos) en una serie de subobjetos: por ejemplo un objeto para el chasis y otro para cada una de las cuatro ruedas; y a continuación puede sacarse partido del hecho de que varios de esos subobjetos son iguales, y emplear una única definición gráfica de rueda en lugar de cuatro.

Otra técnica comúnmente utilizada es la de la descomposición de objetos articulados. Una figura humana, por ejemplo, puede considerarse compuesta de cabeza, tronco y extremidades. Cuando la figura se desplace por la escena, las extremidades adoptarán diferentes posturas, pero es posible que ni la apariencia del tronco ni la de la cabeza sufran variación. En consecuencia, puede descomponerse la figura en varios subobjetos, y emplear un diagrama de animación separado para cada uno. Puesto que la cabeza y el tronco no sufren variación de forma, habrá una única representación gráfica de estos dos objetos, que se compondrá con cada una de las diferentes representaciones gráficas correspondientes a las extremidades.

Una tercera técnica es la de simulación dinámica. El ejemplo típico es el de una explosión. Una explosión puede representarse mediante un único objeto, que va cambiando su representación gráfica, haciéndose más grande a medida que los fragmentos se separan. Pero, para ahorrar memoria, se pueden utilizar muchos objetos pequeños (posiblemente iguales muchos de ellos) y desplazar esos múltiples objetos mediante una simulación de la onda expansiva.

6.2 ARQUITECTURA HARDWARE

6.2.1. Arquitectura básica

Una posible arquitectura hardware, muy simple, para la estación de proceso de un sistema de RV es la mostrada en la figura 6.3.

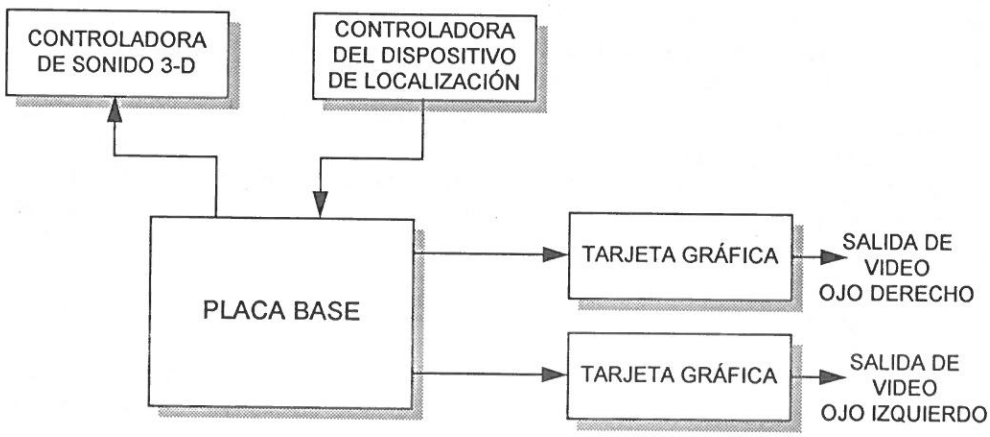

Figura 6.3. Un ejemplo de arquitectura hardware

En esta configuración, a la placa base, que alberga el procesador principal, se conectan los distintos módulos necesarios para implementar la interfaz con los dispositivos periféricos: dos tarjetas gráficas, que generan las imágenes en estéreo de la escena, una tarjeta controladora de sonido 3D y otra tarjeta controladora para el dispositivo de localización utilizado.

El tipo de controladores que se utilice dependerá de cada aplicación en concreto. Es posible, por ejemplo, que la tarjeta de sonido 3D se sustituya por otra tarjeta de sonido convencional, o que no se utilice ningún dispositivo de localización, o que el controlador de éste sea externo a la estación de proceso. También puede ser que haya una única tarjeta gráfica, y que el efecto de estereoscopía se consiga por multiplexación temporal. Por otro lado, pueden existir controladores adicionales (para joysticks, plataformas móviles, ...), tarjetas de conversión analógico/digital para captura de datos, o cualquier otro tipo de tarjeta requerido por los periféricos concretos que el sistema esté empleando.

La tarjeta o tarjetas gráficas empleadas en el sistema pueden incluir su propio procesador o procesadores, que se encargarán de realizar la síntesis de la escena. El procesador gráfico mantendrá algún tipo de estructura de datos consistente, básicamente, en una lista de objetos con sus correspondientes posiciones y representaciones gráficas. El procesador principal actualizará dicha lista con los resultados de las simulaciones efectuadas e informará al procesador gráfico de cuándo puede comenzar a sintetizar la escena.

En ocasiones es posible que la tarjeta gráfica utilizada sólo realice parte del proceso de la síntesis. Se puede, por ejemplo, emplear una tarjeta que actúe sólo como procesador ráster, en cuyo caso es el procesador principal el que debe encargarse del procesamiento geométrico. En este caso, los datos que el procesador principal entregará al procesador gráfico son listas de polígonos ya proyectados en pantalla, con los correspondientes colores de los vértices.

6.2.2. Arquitecturas gráficas avanzadas

Las estaciones especializadas en el tratamiento de gráficos 3D emplean dos métodos distintos para acelerar el proceso de síntesis y aumentar la capacidad del sistema gráfico: paralelismo y pipelining.

6.2.2.1. Arquitecturas pipeline

La primera forma de acelerar el proceso de síntesis de la imagen consiste en utilizar varios procesadores, y repartir entre ellos los pasos del algoritmo.

En el esquema de la figura 6.4, que representa una parte del proceso de síntesis, el procesador 1 realiza el clipping de un polígono y entrega el resultado al procesador 2. Éste hace los cálculos de iluminación correspondientes y entrega el resultado al tercer procesador, el cual efectúa la proyección sobre pantalla, etc...

La clave estriba en que, una vez que el primer procesador ha entregado un resultado al siguiente, queda libre para empezar los cálculos relativos al próximo polígono. Así, en un momento cualquiera, mientras que el procesador 3 está proyectando el polígono n, el

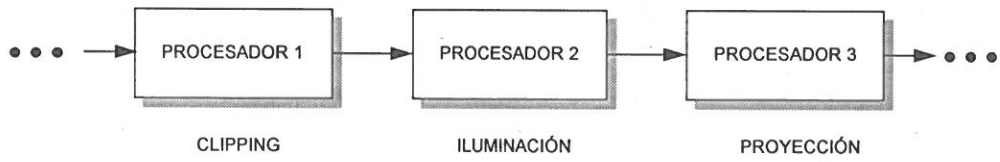

Figura 6.4. Ejemplo de arquitectura pipeline

procesador 2 estará calculando la iluminación del polígono n+1, el procesador 1 estará haciendo el clipping del polígono n+2, etc...

Este método permite ejecutar simultáneamente varios pasos del proceso, con lo que la velocidad total se incrementa.

6.2.2.2. *Arquitecturas paralelas*

La segunda forma de aumentar la velocidad de síntesis consiste en duplicar el número de procesadores dedicados a una tarea específica.

Supongamos, por ejemplo, que nuestro sistema gráfico está dividido en dos módulos: uno dedicado al procesamiento geométrico y otro a la rasterización. Esto de por sí es ya una arquitectura pipeline. El módulo de procesamiento geométrico realizará los cálculos pertinentes para cada polígono e irá entregando el resultado al procesador ráster, el cual se encarga del resto del proceso. Obviamente, ambos procesadores estarán trabajando de manera simultánea: mientras que el procesador ráster trabaja en un cierto polígono, el procesador geométrico irá efectuando los cálculos correspondientes a los polígonos siguientes.

¿Qué sucede si uno de los dos procesadores no tiene capacidad suficiente para adaptarse al ritmo de trabajo del otro? Puede suceder, por ejemplo, que el procesador geométrico tarde mucho menos que el de rasterización en procesar un polígono, en cuyo caso terminará por llenarse el buffer de entrada al procesador ráster y el procesador geométrico se verá forzado a esperar. O puede pasar que quien tarde más sea el procesador geométrico, en cuyo caso el procesador ráster tendrá que permanecer ocioso a la espera de que le entreguen un nuevo polígono que procesar.

En una arquitectura pipeline, la velocidad total del sistema está limitada por el procesador más lento de la cadena.

Una forma de aumentar la velocidad consiste, entonces, en repartir las tareas del paso más lento entre varios procesadores. Supongamos que fuera la etapa de rasterización la más lenta. Entonces se podría aumentar la capacidad del sistema empleando un esquema como el que muestra la figura 6.5.

Aquí, la etapa de rasterización se ha repartido entre dos procesadores, cada uno de los cuales se encarga de una mitad de la pantalla. Cuando el procesador geométrico termina con un cierto polígono, se lo entrega a uno u otro de los dos procesadores ráster, según en qué

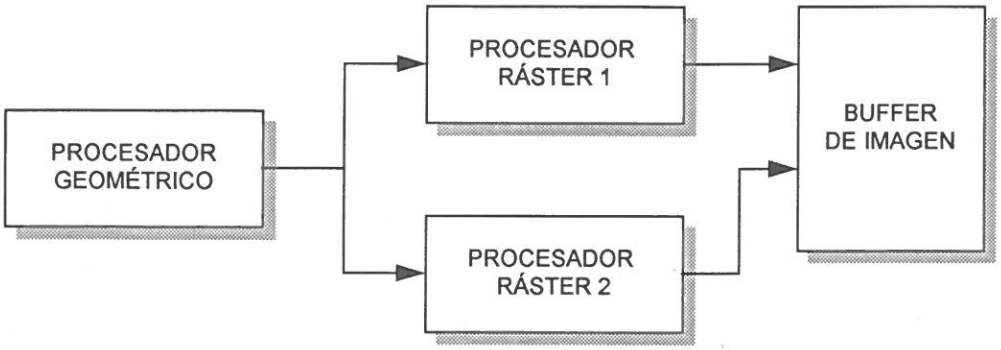

Figura 6.5. Ejemplo de arquitectura paralela

zona de la pantalla caiga el polígono. Si el polígono está a caballo entre las dos, el procesador geométrico lo dividirá en subpolígonos y entregará a cada procesador ráster una parte.

Este tipo de arquitectura se denomina arquitectura paralela, y en ella dos o más procesadores se reparten (de forma más o menos equitativa) el conjunto de datos de entrada, lo que aumenta la velocidad de proceso del sistema.

Observe que cuando se utilizan en un sistema tarjetas separadas para generar las imágenes correspondientes a los dos ojos ya se está, de hecho, utilizando la técnica del paralelismo.

6.2.2.3. Un ejemplo de arquitectura combinada

Ambas técnicas, paralelismo y pipelining, se pueden combinar de cualquier manera que se desee, y las estaciones especializadas en el procesamiento de gráficos 3D suelen permitir diversas opciones de configuración, para adaptar el sistema a las necesidades de cada aplicación concreta.

La estación TWS88 de la empresa francesa Caption, por ejemplo, permite conectar hasta 8 tarjetas de procesamiento geométrico (cada una con 1 a 4 procesadores i860) y 4 tarjetas de rasterización, en cualquier combinación. La figura 6.6 muestra la arquitectura de este sistema.

Según las necesidades de nuestro sistema, dotaremos con más recursos a una u otra parte. La comunicación entre las tarjetas se realiza mediante un doble bus. Las tarjetas de procesamiento geométrico se comunican con el procesador (o procesadores) principal y con las tarjetas de rasterización a través de un bus VME. La tarjeta de rasterización, por su parte, se comunica con la controladora de display mediante un bus especializado.

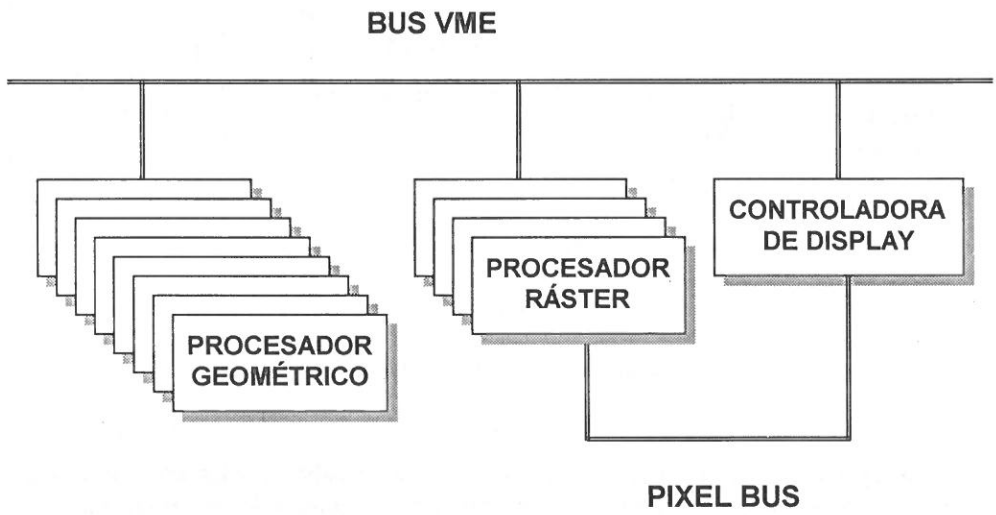

Figura 6.6. Diagrama de la estación TWS88 de Caption

6.2.3. Arquitecturas distribuidas

En ocasiones, los sistemas de RV se configuran alrededor de no una, sino varias estaciones de trabajo.

Hay tres razones para hacer esto. La primera es que no todos los periféricos pueden usarse con cualquier plataforma, ni todas las herramientas software están disponibles para cualquier tipo de estación. Por ejemplo, es posible que se emplee una cierta plataforma especializada en el tratamiento de gráficos 3D para todo lo relacionado con la síntesis de imágenes, pero que no existan controladores de determinados periféricos para la misma. En este caso, puede emplearse una segunda estación para las tareas de control de E/S, la cual se puede comunicar con la estación gráfica a través de una red de área local.

La segunda razón es que, en ocasiones, los sistemas de RV se construyen sobre sistemas preexistentes, donde es posible que ya estén desarrollados ciertos programas o procedimientos que la aplicación de RV necesite. Puede suceder, por ejemplo, que ya exista una serie de algoritmos de simulación dinámica implementados en una cierta plataforma, y que lo único que se pretenda es extender el sistema, dotándolo de la posibilidad de visualizar los datos en forma de gráficos 3D. Si la plataforma original no se adapta bien a la tarea gráfica, pueden hacerse dos cosas: reescribir los algoritmos de simulación en la nueva plataforma gráfica, o conectar ambas plataformas mediante algún tipo de red.

La tercera razón es que podemos aumentar el rendimiento del sistema empleando más de una plataforma, aplicando la técnica del pipelining o del paralelismo.

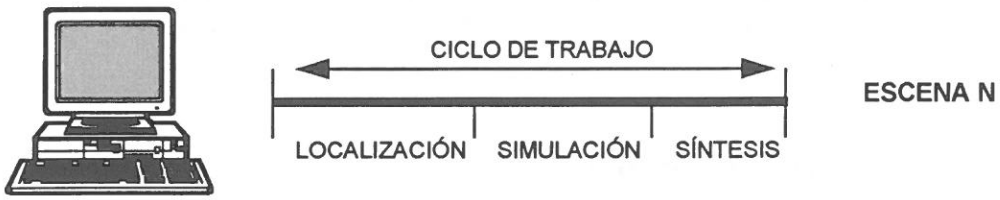

Figura 6.7. Ciclo de trabajo en un sistema monomáquina

Podemos, por ejemplo, dividir las diferentes tareas entre distintas estaciones, que funcionan simultáneamente. La figura 6.7 muestra lo que sucede en el caso de tener una única plataforma. El ciclo de trabajo vendrá dado por la suma de los tiempos invertidos en cada una de las tareas: localización, simulación y síntesis (consideramos sólo en el ejemplo aquéllas que más tiempo consumen). Es posible que las tareas se realicen secuencialmente, como muestra la figura o (si tenemos una estación con un sistema operativo multitarea) de manera entremezclada. Pero, en cualquier caso, será la suma de esos tiempos la que determine el número máximo de ciclos por segundo que el sistema podrá realizar.

Sin embargo, si utilizamos varias estaciones, como muestra la figura 6.8, a cada una de las cuales le encargamos de una de las tareas, el ciclo de trabajo se reducirá, y vendrá determinado por la tarea que más tiempo consuma (en este caso la localización). Con este reparto de tareas, mientras que una de las plataformas esté haciendo la síntesis de la escena n, otra de ellas estará realizando los cálculos de simulación correspondientes a la escena n+1, y la tercera estará capturando los datos de posición del usuario correspondientes a la escena n+2.

Observe un dato importante: el pipelining permite aumentar el número de ciclos totales por segundo, porque disminuye el ciclo de trabajo, **pero no disminuye la latencia**. Es decir, si nos fijamos en lo que sucede en una escena determinada, n, el tiempo total transcurrido desde que se empiezan a adquirir los datos de localización, hasta que se realiza la síntesis de **esa** escena no disminuye, sino que continúa siendo la suma de los tiempos de cada una de las tareas parciales.

Puede incluso suceder que la latencia del sistema aumente, debido a los retardos de comunicación adicionales que la red de área local introduce. En concreto, una Ethernet puede introducir retardos adicionales en el procesamiento de hasta 40 mseg. En consecuencia, a la hora de realizar el diseño del sistema, hay que ponderar cuidadosamente la latencia total que se derive de la arquitectura que elijamos.

También se pueden emplear soluciones basadas en el paralelismo para disminuir el ciclo de trabajo. Existen paquetes software comerciales de síntesis de imágenes, como Renderman por ejemplo, que permiten distribuir de manera transparente las tareas de síntesis entre diversas máquinas. Para el caso de la simulación, sin embargo, no existen aún soluciones

similares, por las dificultades que plantea mantener la coherencia de la simulación cuando ésta se reparte entre procesadores distintos. Hablaremos de ello en el capítulo 8.

6.3. TIPOS DE ESTACIONES

Las estaciones de proceso utilizadas comúnmente en aplicaciones de realidad virtual pueden clasificarse en tres distintas categorías.

- Estaciones especializadas.
- Estaciones gráficas.
- Estaciones de propósito general.

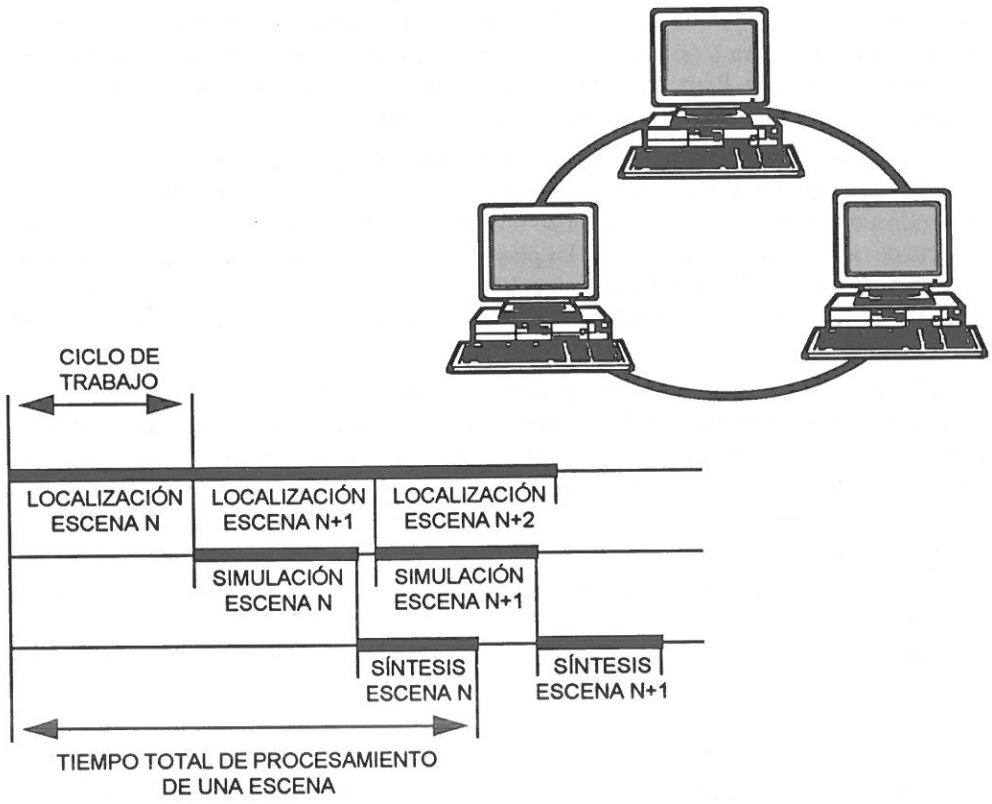

Figura 6.8. Ciclo de trabajo utilizando una arquitectura pipeline

6.3.1. Estaciones especializadas

Son estaciones cuya arquitectura hardware/software ha sido desarrollada pensando específicamente en las necesidades de las aplicaciones de realidad virtual y simulación. De entre las empresas que fabrican este tipo de estaciones podemos destacar a la firma americana Evans & Sutherland, una de las empresas pioneras del campo de la realidad virtual, y a las empresas británicas Division y W Industries.

Evans & Sutherland fabrica distintos modelos, desde los de gama baja ESIG-200 y ESIG-500, utilizados en simuladores de vuelo de bajo coste, hasta las series ESIG-2000, ESIG-3000 y ESIG-4000, de aplicación a sistemas de entrenamiento militar, entretenimiento y simuladores de vuelo de alta calidad. Las diferentes estaciones de la gama tienen distintas capacidades gráficas que van desde las funciones básicas de soporte de texturado, anti-aliasing y sombreado, con una velocidad de 60 imágenes por segundo, hasta funciones avanzadas como la gestión automática de hasta 12 niveles de detalle.

Evans & Sutherland tiene una importante cuota de mercado dentro del campo de la simulación profesional, y en la actualidad busca expandir su mercado en dos direcciones: hacia el mundo de los sistemas de realidad virtual de bajo coste, con el lanzamiento de la serie Liberty de estaciones de simulación; y hacia el mercado del ocio, para el que están desarrollando, junto con Iwerks Entertainment, una serie de simuladores de ocio interactivos, denominados Virtual Adventures.

Evans & Sutherland ha establecido también diversos acuerdos con el fin de explotar su tecnología de tratamiento de gráficos 3-D en otros entornos. Así, ha desarrollado una serie de aceleradores gráficos para las estaciones Sun SPARCstation y ha firmado un acuerdo con la firma japonesa Namco para el desarrollo de máquinas para galerías de videojuegos.

La empresa Division, por su parte, fabrica las series Provision 100 y Provision 200 para, respectivamente, uno o dos participantes. Ambas series constan de dos modelos distintos, el VRX (estándar) y el VTX (con texturado). Son estaciones basadas en un 486 con sistema operativo Unix System V. Vienen equipadas con visiocasco, tarjeta de audio tridimensional Convolvotron, de la empresa Crystal River Engineering, y con el entorno software de ejecución y desarrollo de la propia Division. Aunque Division es una empresa británica, los aceleradores gráficos de sus estaciones han sido desarrollados en la Universidad de Carolina del Norte.

La empresa W Industries, fabricante de sistemas recreativos de RV, utiliza también sus propias estaciones especializadas, denominadas Virtuality 1000 y Virtuality 2000. Recientemente, W Industries e IBM han firmado un acuerdo para desarrollar una nueva estación de desarrollo de aplicaciones de RV, el Proyecto Elysium. El sistema utilizará los aceleradores gráficos y el software de desarrollo de aplicaciones de RV de Virtuality sobre una estación IBM ValuePoint/DX4 50/100.

6.3.2. Estaciones gráficas

Son estaciones cuya arquitectura hardware/software ha sido desarrollada pensando en las aplicaciones de tratamiento de gráficos 3-D, lo cual incluye no sólo el mundo de la realidad

virtual y simulación, sino también las aplicaciones de animación profesional, CAD y visualización científica.

La firma americana Silicon Graphics, especializada desde su creación en el diseño y fabricación de estaciones de trabajo para aplicaciones gráficas 3D, es sin lugar a dudas el líder del mercado. Prácticamente todos los paquetes de software gráfico corren sobre las estaciones Silicon Graphics. Últimamente, los fabricantes de estaciones de propósito general, como Sun y Digital, han empezado a introducirse en el mercado de las aplicaciones gráficas mediante tarjetas gráficas aceleradoras desarrolladas por terceras fuentes. Ante esto, y ante la competencia cada vez mayor de los sistemas basados en PC, Silicon Graphics ha reaccionado intentando estandarizar "de facto" la interfaz software Open GL, basada en la librería gráfica GL de sus estaciones de trabajo.

Las estaciones Silicon Graphics incorporan diversos tipos de subsistemas gráficos. La última generación de plataformas emplea el subsistema gráfico Reality Engine, cuya configuración máxima incluye 353 procesadores, con una velocidad de 240 millones de pixels texturados por segundo y 1 millón de triángulos texturados con antialiasing por segundo.

La gama de estaciones de Silicon Graphics es bastante amplia, tanto en prestaciones como en precios. En la parte alta se encuentran las estaciones de las series Onyx y Crimson, mientras que el segmento más bajo lo cubren las series Indigo e Indy. Mientras que una plataforma Silicon Graphics de gama alta puede sobrepasar los 40 millones de pesetas, el precio de la estación Indy puede ir de 1 a 4 millones de pts. La Indy tiene funciones de sombreado, texturado, alfa-blending y anti-aliasing, además de soportar aplicaciones estereoscópicas. Su velocidad máxima de síntesis va de 130K a 200K polígonos (con sombreado de Gouraud) por segundo, según modelo.

La Indigo 2 tiene una arquitectura más potente, con hasta 8 procesadores geométricos y 2 procesadores ráster. Ofrece todas las funcionalidades de la Indy y además: z-buffer de 32 bits, hasta 8 fuentes de luz, iluminación difusa y especular y fuentes de luz puntuales. Su precio va de 3,5 a 8 millones de pts, y su velocidad máxima de síntesis es de 245K a 650K polígonos por segundo.

A nivel europeo, la empresa Caption, compañía francesa del Grupo Telmat, desarrolló la estación gráfica TWS88 a partir de la investigación realizada dentro de un proyecto europeo ESPRIT. Tiene sistema operativo Unix SVR4 y arquitectura multiprocesador basada en el bus VME. La interfaz gráfica 3D es PHIGS, con protocolo PEX, y la librería gráfica de bajo nivel es una librería propietaria, denominada GIL.

Tiene una capacidad de proceso máxima de 150 Mips por tarjeta procesadora (admite un máximo de cuatro tarjetas), y permite generar hasta 805K polígonos/segundo con sombreado de Gouraud, z-buffering de 32 bits, y 24 bits color. La máxima resolución que soporta es de 1536x1152 pixels.

6.3.3. Estaciones de propósito general

El tercer grupo de plataformas utilizadas en aplicaciones de realidad virtual está constituido por estaciones de propósito general: las estaciones SPARC de Sun, las estaciones

Alpha de Digital o las estaciones basadas en PC. Estas estaciones poseen una arquitectura que no está pensada para el tratamiento de gráficos 3D, por lo que necesitan que se les añada el necesario hardware gráfico de aceleración.

Existen diversas compañías que fabrican tarjetas gráficas para una u otra de estas arquitecturas. Así, Evans & Sutherland ha desarrollado una serie de aceleradores gráficos para Sun, denominados Freedom Series, que proporcionan hasta 3 Mpolígonos por segundo y soportan como APIs XGL, SunPHIGS, Xlib, PEXlib y la GL de Silicon Graphics. La firma DuPont Pixel fabrica también distintas tarjetas aceleradoras para Sun, que permiten ejecutar muchas de las aplicaciones desarrolladas para Silicon Graphics.

Para las estaciones Alpha AXP de Digital hay disponibles aceleradores gráficos, como el Denali E15 de Kubota, con funciones de sombreado, z-buffer de 24 bits, anti-aliasing, luces puntuales y capacidad estereoscópica. Alcanza una velocidad de 50K polígonos por segundo. Sin embargo, Kubota ha anunciado recientemente su intención de discontinuar la producción de la serie Denali, para dedicarse a los aceleradores gráficos para PC.

Hewlett Packard ha anunciado también un acuerdo para incorporar los aceleradores gráficos de la serie Freedom de Evans & Sutherland en sus estaciones HP 9000. Los aceleradores utilizarán una interfaz Open GL. Asimismo ha establecido un acuerdo de colaboración con la empresa Division, que prevé la construcción de estaciones de proceso para aplicaciones de RV basadas en las HP 9000 Serie 700. Las estaciones incorporarán el acelerador gráfico de las estaciones Provision 100 VRX, así como el software de desarrollo y el entorno de ejecución de Division.

Pero el mercado más promisorio está en el campo de las estaciones basadas en PC, dado que cada vez están apareciendo más tarjetas gráficas para este tipo de entorno, que permiten construir aplicaciones de realidad virtual a un coste mucho menor que las soluciones basadas en otras plataformas.

La mayoría de las tarjetas existentes para PC permiten alcanzar resoluciones de hasta 1280x1024 bits, con una paleta de colores de 24 bits y sombreado de Gouraud. Todas ellas tienen doble salida de vídeo o, al menos, permiten conectar dos tarjetas en paralelo, para aplicaciones con visión estereoscópica. La velocidad de síntesis de los distintos modelos existentes en el mercado es muy variable. Entre las más extendidas podemos destacar:

- La tarjeta Fire 860, de la compañía alemana Spea, muy utilizada en multitud de aplicaciones, ya que fue de las primeras en salir al mercado. Utiliza un procesador i860 de Intel. Sus capacidades gráficas son relativamente escasas, ya que proporciona una velocidad de síntesis de 8.500 polígonos por segundo, con sombreado de Gouraud, o 7.500 polígonos con textura. Dispone de 6 MBytes para memoria de texturas, ampliables a 14 MB.
- dVIEW, fabricada y comercializada por la compañía británica Division en versión PC-ISA y Macintosh. Tiene salidas NTSC y VGA, y una velocidad de síntesis de 35.000 polígonos y 16 Mpixels por segundo. Al igual que la Fire 860, utiliza un procesador i860, y también permite utilizar texturas.

- Stride, una tarjeta para bus ISA de la empresa Primary Image, con una velocidad de síntesis de 35.000 polígonos y 5 Mpixels por segundo. Utiliza el procesador TMS34020 de Texas, y tiene interfaz TIGA.

6.4. ARQUITECTURA SOFTWARE

6.4.1. Diagrama general

La arquitectura software típica de un sistema de RV se muestra en la figura 6.9.

Además del sistema operativo de la plataforma elegida, el sistema incorporará una serie de drivers o librerías para el control de los dispositivos de entrada/salida utilizados. Dichos drivers se presentarán, normalmente, como librerías en C (u otro lenguaje), que pueden añadirse a la aplicación en tiempo de compilación. Más adelante hablaremos acerca de las interfaces gráficas más populares.

Tanto el sistema operativo como los drivers y librerías gráficas utilizados tienen como misión liberar a la aplicación de las tareas asociadas al control directo del hardware, así como implementar determinados algoritmos especializados (generación de gráficos, generación de sonido 3D, localización) cuyos detalles de funcionamiento no tienen por qué ser

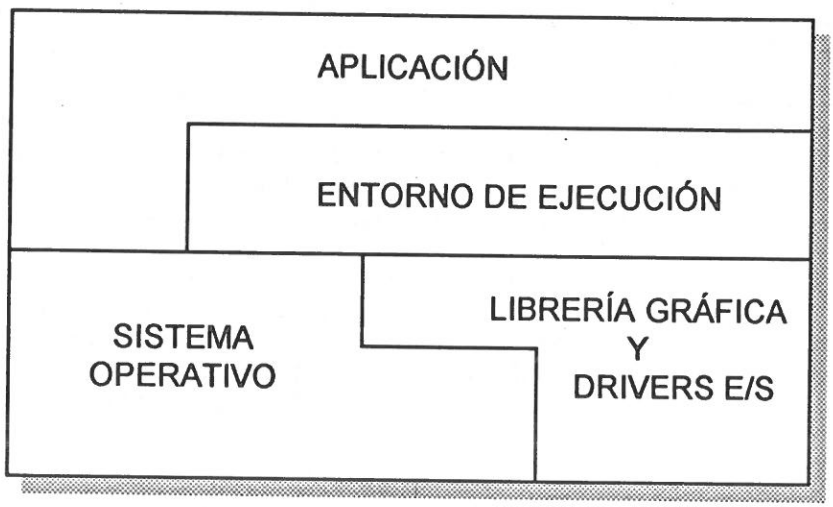

Figura 6.9. Arquitectura software de la estación de trabajo

conocidos por el programador de aplicaciones. Dichos algoritmos estarán, normalmente, optimizados para el hardware concreto que se esté utilizando.

6.4.2. Librerías gráficas

Como ya hemos dicho, es muy posible que el programador de aplicaciones de RV no necesite conocer en absoluto los detalles de la mayoría de los drivers utilizados en su sistema, incluyendo la librería gráfica encargada de la síntesis de las escenas. Para ello, es necesario que el entorno de ejecución utilizado se encargue de proveer esa transparencia.

Hay ocasiones, sin embargo, en que es necesario utilizar de manera directa los drivers de nuestro hardware gráfico. Existen circunstancias, por ejemplo, en las que no es posible utilizar ningún entorno de ejecución comercial, porque ninguno de los existentes se adapta a las necesidades de nuestro sistema. Es posible que no haya ningún entorno adecuado que corra sobre la plataforma elegida, o que los entornos existentes para dicha plataforma no proporcionen la funcionalidad requerida.

En estos casos, el programador puede verse forzado a implementar él mismo todo el proceso de síntesis de la imagen o parte del mismo, para lo cual deberá usar las distintas funciones que la librería gráfica seleccionada proporcione.

Existen dos interfaces gráficas estándar:

- GKS-3D (Graphical Kernel System - 3D), estándar ISO-8805. Versión 3D del anterior estándar GKS, que estaba orientado a aplicaciones 2D. El estándar define un conjunto de funciones gráficas para la síntesis de escenas tridimensionales. El estándar está definido de forma que sea independiente del sistema operativo empleado, del lenguaje de desarrollo o de la plataforma utilizada. Tiene la desventaja de que las facilidades que ofrece para la estructuración de las escenas no son muy adecuadas.
- PHIGS+ (Programmer's Hierarchical Interactive Graphics System), estándar ISO-9592. Más orientado a aplicaciones interactivas que GKS-3D y con más facilidades de estructuración. Comparte con GKS-3D la independencia con respecto a la máquina, el lenguaje y el sistema operativo. Se usa principalmente en estaciones gráficas de gama alta, basadas en sistema operativo Unix.

Estas interfaces estándar presentan un problema fundamental. El proceso de síntesis de las escenas es un elemento crítico de los sistemas gráficos, por lo que es necesario optimizarlo para sacar el máximo rendimiento del hardware utilizado, y para adaptarlo a cada tipo concreto de aplicación. Las optimizaciones del proceso de síntesis que es posible realizar dependen sobremanera de cada aplicación concreta y de las capacidades que el hardware proporcione, por lo que una interfaz genérica como las dos mencionadas suele proporcionar soluciones que distan bastante de la óptima.

Por poner un ejemplo, una misma estación gráfica puede utilizarse para aplicaciones de CAD, para ejecutar un programa de animación comercial, o para soportar una aplicación de RV. Todas estas aplicaciones comparten una necesidad común: la generación de los gráficos a la más alta velocidad posible. Pero cada una de ellas pone el acento en un aspecto funcional

diferente: la aplicación de CAD requerirá principalmente una serie de capacidades de edición interactiva, la de animación buscará generar escenas de muy alta calidad, y la de realidad virtual necesitará generar cada escena en un lapso de tiempo máximo prefijado.

Disponer de una interfaz estándar permitiría que el desarrollador del programa de CAD, del programa de animación y de la aplicación de RV programaran sus respectivas aplicaciones de una manera sencilla; sin embargo, obtendrían soluciones muy poco optimizadas en cuanto a velocidad de generación de imágenes. Pero si dispusieran de una forma de adaptar el proceso de síntesis a su aplicación concreta, podrían obtener un rendimiento muy superior, aunque fuera a costa de una labor de programación mayor.

Por ello ninguno de los dos estándares tiene una gran aceptación y la mayoría de los fabricantes tienden a proporcionar librerías gráficas propietarias de bajo nivel, las cuales proporcionan un control mucho más profundo del proceso de síntesis. Los entornos de ejecución de aplicaciones de RV comerciales utilizan estas librerías para proporcionar los servicios de alto nivel que ponen a disposición de las aplicaciones. Si alguna vez el programador de aplicaciones necesita implementar alguna función de síntesis, u optimizar las existentes, este tipo de control a bajo nivel le permite hacerlo de acuerdo a las necesidades concretas de su aplicación.

Así, Silicon Graphics proporciona su librería Iris GL, Sun proporciona la XGL y HP la Starbase. En el campo del PC, cada fabricante de tarjetas proporciona sus propios drivers o se adapta a las especificaciones dictadas por los entornos de ejecución más populares, como WorldToolkit o SuperScape.

Recientemente, Silicon Graphics, ante el empuje de las soluciones basadas en PC, ha contraatacado tratando de estandarizar "de facto" un derivado de su Iris GL, denominado Open GL. Junto con Digital, IBM, Microsoft e Intel, ha formado un consorcio que se encarga de otorgar las oportunas licencias a los fabricantes de hardware, entornos de desarrollo y sistemas que deseen incorporar Open GL en sus productos. Open GL está pensada para funcionar tanto sobre estaciones gráficas y estaciones de trabajo de propósito general, con sistema operativo Unix y sistema de ventanas X-Windows, como sobre PCs, utilizando Windows como sistema de ventanas. Posee funciones para iluminación, sombreado, z-buffering, anti-aliasing, texturado, etc... En principio está previsto proporcionar bindings para C, C++, Fortran y Ada.

En caso de tener éxito en su intento de estandarizar Open GL, Silicon Graphics conseguiría forzar a los fabricantes de tarjetas gráficas para PC a proporcionar dicho tipo de interfaz, con lo que las aplicaciones podrían ser fácilmente portadas de estaciones Silicon Graphics a PC, y viceversa. Aparte, claro está, de los ingresos por licencias, que no van a parar a Silicon Graphics sino al consorcio Open GL.

6.4.3. Entorno de ejecución

Desde el punto de vista del programador de aplicaciones de realidad virtual, no basta con ocultar los detalles de la implementación hardware mediante un conjunto de librerías, drivers y servicios del sistema operativo. La complejidad de las aplicaciones de RV requiere

de una interfaz de más alto nivel, en la que el programador pueda manejar los objetos que componen el mundo virtual de manera directa, despreocupándose de la interfaz software que cada dispositivo concreto utilice.

Por eso las aplicaciones de realidad virtual tienden cada vez más a emplear una capa intermedia adicional, el entorno de ejecución, en la que se implementan todos los mecanismos básicos de manejo de objetos. Cada sistema de desarrollo de aplicaciones de realidad virtual, como SuperScape o WorldToolKit (de los cuales hablaremos en el próximo capítulo) incluye su propio entorno para la ejecución de aplicaciones de RV. El entorno de ejecución soportará, normalmente, funciones como:

- Inicialización del mundo virtual y de los objetos que lo componen.
- Desplazamiento, rotación o cambio de escala de un objeto dentro del mundo virtual.
- Detección de colisiones.
- Soporte de animaciones y secuencias de sonido, incluyendo las cuestiones relacionadas con la sincronización.
- Activación, desactivación y movimientos de las fuentes de luz.
- Movimientos de las cámaras.
- Entrada y salida de datos.

El entorno de ejecución puede también soportar funciones más sofisticadas, como la conmutación del nivel de detalle, o la variación dinámica de los algoritmos de síntesis (control de complejidad).

Además, el entorno de ejecución suele encargarse de controlar todos los dispositivos del sistema, de tal manera que el programador de aplicaciones no tenga ni tan siquiera que conocer los drivers utilizados. Sólo será necesario utilizar de modo directo algún driver cuando se utilice algún periférico que el entorno de ejecución escogido no soporte.

Sobre estas facilidades que el entorno de ejecución proporciona, el programador de aplicaciones debe construir el código correspondiente a la simulación del mundo virtual, implementando las funciones que determinen el comportamiento de los objetos, las interacciones entre los mismos y la gestión de cualesquiera variables globales que el sistema utilice. Algunos entornos de ejecución, como CDK de Autodesk, proporcionan algunas facilidades para la definición de determinados comportamientos simples de los objetos. En concreto, CDK tiene ya predefinidas determinadas leyes físicas, como la gravedad o la inercia.

La forma que el entorno de ejecución adopte será diferente según el tipo de plataforma que estemos utilizando. Así, en los sistemas multiproceso, por ejemplo las estaciones basadas en Unix, el entorno de ejecución puede ser un proceso independiente con el que la aplicación se comunicará. En otros sistemas, el entorno de ejecución puede estar implementado en forma de librería, que se linca con la aplicación. En otros sistemas, finalmente, la aplicación puede no existir como tal código, sino consistir en una serie de tablas que el entorno de ejecución emplea.

7
Desarrollo de aplicaciones de RV

A lo largo de los capítulos anteriores hemos pasado revista a los distintos elementos hardware y software que forman parte de un sistema de RV, hemos visto cuáles eran los principales factores que determinan el rendimiento de un sistema, y hemos expuesto los principios en que se basa el funcionamiento de las aplicaciones.

En este capítulo, vamos a analizar cuál es el proceso de desarrollo de una aplicación, desde el momento en que se determina el objeto de la misma hasta que se realiza la integración final de nuestro mundo virtual.

Presentaremos, en primer lugar, un esquema global del proceso de desarrollo, para luego analizar en detalle cada uno de los pasos que lo componen, resaltando los puntos críticos y las herramientas más comúnmente utilizadas. Finalmente, presentaremos los entornos más populares de desarrollo de aplicaciones de RV, con sus características fundamentales.

7.1. CÓMO SE CREA UN PROGRAMA DE RV

En el proceso de desarrollo de un sistema de RV podemos diferenciar dos grandes etapas: diseño del sistema y desarrollo de la aplicación. En la primera etapa, será necesario diseñar (si es que no existe ya) el sistema sobre el que la aplicación va a ser ejecutada, seleccionando e integrando los distintos elementos que lo van a componer.

Básicamente, procederemos a estudiar en primer lugar cuáles son los requerimientos que la aplicación que se va a desarrollar impone a nuestro sistema, para después proceder a la construcción de éste a partir de los componentes hardware y software existentes comercialmente.

En la segunda etapa, la de desarrollo de la aplicación, procederemos a construir el mundo virtual, a partir de las facilidades que el sistema pone a nuestra disposición. Primero será necesario diseñar la estructura básica del mundo virtual, definiendo la estructura de la escena (o escenas), identificando los objetos que la componen y seleccionando para ellos la apariencia y el comportamiento deseados. Después, deberemos proceder a construir cada uno de dichos objetos.

Básicamente, construir un objeto consta de dos procesos separados: el proceso de desarrollo artístico, mediante el que se crean las representaciones gráficas y auditivas de los

objetos y de las escenas, y el proceso de desarrollo funcional, mediante el que se define el comportamiento de los objetos y la mecánica general de la aplicación.

El comportamiento de los objetos vendrá determinado por cada aplicación en particular. Mientras que en un simulador de vuelo las acciones de los objetos se traducen en desplazamientos a través del mundo virtual, una aplicación científica de visualización de turbulencias en un túnel de viento puede requerir implementar complejas leyes matemáticas de la mecánica de fluidos.

Normalmente, para la definición de la apariencia (gráfica, sonora, ...) de los objetos emplearemos herramientas software especializadas, tales como programas de animación o programas de creación de efectos sonoros. En cuanto al comportamiento de los objetos, se definirá en algún lenguaje de programación, como C o C++, utilizando las librerías puestas a nuestra disposición por el entorno de ejecución de aplicaciones de RV seleccionado y por los drivers de los periféricos de RV empleados.

Una vez construidos los objetos, procederemos finalmente a implementar el mundo virtual, definiendo el estado inicial de los objetos para cada una de las escenas que lo componen, definiendo las variables globales del mundo virtual y sus mecanismos de actualización y, en definitiva, integrando los distintos objetos que forman parte del mundo virtual. Durante esta fase, lo más probable es que se realicen una serie de ciclos iterativos para ajustar las distintas características de los objetos, refinando su aspecto o su comportamiento hasta conseguir que el funcionamiento global del sistema sea el deseado.

La Figura 7.1 muestra los pasos básicos en que se divide el proceso global de desarrollo de un sistema de RV.

7.2. ESPECIFICACIÓN DE REQUERIMIENTOS

Una vez que se conoce con precisión cuál es el objetivo último de la aplicación que se va a desarrollar, lo primero será determinar el tipo de sistema necesario y sus características fundamentales. ¿Es necesario un sistema inmersivo, o se trata de una aplicación de sobremesa? ¿Es necesaria la estereoscopía? ¿Cuántos usuarios simultáneos va a tener el sistema?...

Básicamente, de lo que se trata es de definir las características que el sistema debe poseer, con el mayor detalle posible. Algunos de los factores que habrá que analizar son los siguientes:

1) ¿Qué tipo de sistema es necesario: inmersivo, proyectivo o de sobremesa?
2) ¿Se trata de un sistema monousuario o multiusuario?
3) ¿De qué presupuesto total se dispone para el diseño del sistema y el desarrollo de la aplicación? ¿De cuánto tiempo se dispone? ¿De qué personal?
4) Una vez que la aplicación esté en marcha, ¿cuáles son los costes máximos permisibles de uso?
5) ¿Existen facilidades hardware o software preexistentes que tengamos que utilizar? Si es así, ¿cuáles son éstas? Si no, ¿existe algún tipo de preferencia en cuanto a la

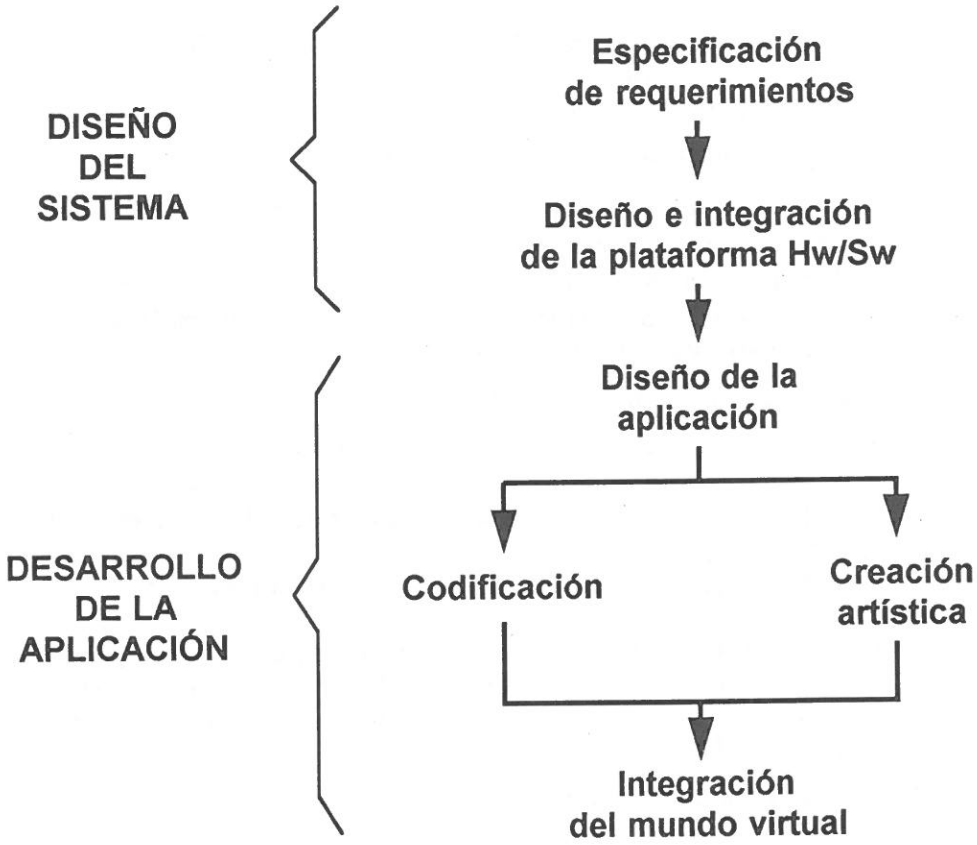

Figura 7.1. El proceso de desarrollo de un sistema de RV

plataforma que se utilizará, en cuanto a los periféricos o en cuanto a las herramientas software?

6) ¿Cuál es el objetivo exacto de la aplicación, es decir, qué va a hacer el usuario con ella? ¿Qué modos de interacción hay que poner a disposición del usuario?
   - ¿Qué tipo de objetos va a contener el mundo virtual?
   - ¿Qué va a poder hacer el usuario con ellos?
   - ¿Qué facilidades de navegación se van a poner a disposición del usuario? ¿Existen restricciones al movimiento dentro del mundo virtual?
   - ¿Cuál es el máximo valor permisible para la latencia total del sistema?
   - ¿Necesitará el sistema detectar de manera automática la orientación del usuario? ¿Y la posición?

- El usuario, ¿va a estar sentado mientras utiliza la aplicación o va a moverse libremente en una cierta área?
- ¿Durante cuánto tiempo seguido va a emplear el usuario la aplicación cada vez?

7) ¿Qué características tiene el usuario potencial?

- ¿Poseen los futuros usuarios algún conocimiento o entrenamiento especiales?
- ¿Tienen alguna peculiaridad física?
- ¿Pertenecen a algún grupo de edad concreto?
- ¿Existirá alguna limitación de uso del sistema, como prohibición de uso a menores o a personas con alguna minusvalía física?

8) ¿Qué requerimientos debe poseer el sistema gráfico?

- ¿Es necesaria la visión estereoscópica? ¿Es deseable?
- ¿Cuál es la resolución mínima requerida? ¿Cuál es la deseable?
- ¿Cuál es velocidad mínima admisible de generación de las imágenes, en número de imágenes por segundo?
- ¿Cuántos objetos va a contener una escena del mundo virtual, por término medio? ¿Y como máximo?
- ¿Cuál es la calidad mínima permisible de las imágenes? ¿Cuántos polígonos se utilizarán como mínimo para definir cada objeto? ¿Y por término medio?
- ¿Existe algún tipo de requerimiento en cuanto al modelo de iluminación que se va a utilizar?

9) ¿Qué otro tipo de información se va a proporcionar al usuario, además de la puramente gráfica?

- ¿Es necesario el sonido? ¿Es necesario proporcionar mecanismos de síntesis de sonido o basta con reproducir sonidos pregrabados?
- ¿Hace falta sonido tridimensional?
- ¿Hacen falta mecanismos de síntesis de voz?
- ¿Se va a proporcionar al usuario realimentación táctil o cinestésica?
- ¿Debe incluir el sistema algún tipo de elemento móvil, como plataformas hidráulicas?
- ¿Necesitará el sistema controlar algún otro tipo de equipo externo, como brazos mecánicos o maquinaria?

10) ¿Qué requerimientos de simulación existen?

- ¿Hay algún programa de simulación preexistente con el que sea necesario construir una interfaz?
- ¿Existe alguna estimación de los recursos de CPU necesarios para la simulación?
- ¿Qué precisión requieren las simulaciones en el tratamiento de los datos? ¿Cuántos ciclos de simulación son necesarios por segundo?

11) ¿Cómo va a controlar el usuario el funcionamiento del sistema?

- ¿Hacen falta dispositivos especiales para controlar la aplicación, tal como electroguantes o ratones 3D, o basta con dispositivos convencionales?
- ¿Es necesario utilizar algún dispositivo mecánico concreto, por ejemplo para simular los controles de un vehículo real?
- ¿Va a tener el usuario algún impedimento de carácter físico a la hora de emplear el sistema? ¿Va a poder usar las manos con libertad?

12) ¿De qué herramientas de desarrollo se dispone?

- Programas de animación y de diseño de gráficos 3D.
- Programas de diseño de gráficos 2D.
- Equipos de digitalización.
- Programas de composición de secuencias musicales y efectos sonoros.
- Ordenadores disponibles para desarrollo. ¿Qué sistemas operativos utilizan?
- Compiladores.

13) ¿Va a ser necesario entrenar a los desarrolladores en las herramientas utilizadas? Si es así, ¿pueden evaluarse el tiempo y el coste de ese entrenamiento?
14) ¿Va a ser necesario entrenar a los usuarios de la aplicación una vez desarrollada? Si es así, ¿puede evaluar el tiempo y el coste del entrenamiento?
15) ¿Van a desarrollarse los aspectos artísticos en la empresa o se van a subcontratar? En el primer caso, ¿se dispone realmente del personal necesario para obtener la calidad artística requerida?

Lo más posible es que muchas de estas preguntas no puedan responderse, o sólo pueda hacerse de manera aproximada. Hay aplicaciones, por ejemplo, que requieren forzosamente un sistema inmersivo, mientras que otras pueden ser de sobremesa por su propia naturaleza. Pero hay situaciones en las que tan válida es una solución como otra. En estos casos, la decisión puede aplazarse hasta más adelante, comparándose las ventajas y desventajas de cada solución con los costes y con los factores de disponibilidad de componentes.

También es posible que determinados parámetros, como el número de objetos que poblarán la aplicación, o el número de polígonos de que éstos estarán compuestos, no se conozcan a priori. Pero lo que siempre se podrá establecer son unos límites máximos y mínimos, para obtener unas indicaciones sobre la importancia relativa de los distintos factores y su "bondad" con respecto al coste.

Entre todos los factores mencionados, el de los tipos de interacción puestos a disposición del usuario, es decir, la especificación funcional del sistema, tiene una especial relevancia. La forma de interacción requerida determinará, posiblemente, algunas características de los periféricos necesarios, y ayudará a comprender qué es importante y qué no en la aplicación.

Al final, lo que se obtendrá será una descripción más o menos detallada del sistema que se construirá, en la que posiblemente queden aún muchas preguntas sin responder, pero en la que al menos se habrán trazado los límites que el sistema no puede traspasar.

7.3. DISEÑO E INTEGRACIÓN DE LA PLATAFORMA HW/SW

Si durante la fase de especificación de requerimientos se realiza una pintura abstracta del sistema, en esta segunda fase esa pintura se perfila y concreta, respondiendo a cada una de las preguntas que quedaron anteriormente sin contestar.

En esta etapa se realiza el diagrama de bloques del sistema, y se comprueban los costes, disponibilidad de componentes, compatibilidad de las herramientas, etc... También se lleva a cabo un análisis de hasta qué punto cumple el sistema con los requerimientos de operación. Dependiendo de las decisiones que en esta fase se tomen, muchas de las preguntas que anteriormente quedaron sin contestar acerca de las características del sistema serán respondidas, y la capacidad de éste quedará determinada.

La selección de los componentes abarca todos los aspectos del sistema sobre el que la aplicación se va a ejecutar: plataforma o plataformas hardware, sistema operativo, aceleradores gráficos, periféricos, entorno software de ejecución; pero también es necesario proceder a seleccionar los distintos componentes del sistema de desarrollo: plataforma, herramientas de animación, compiladores, ...

Con respecto al tema de la compatibilidad entre componentes, algunos factores obvios que es necesario comprobar son:

- ¿Sobre qué plataforma y sistema operativo funciona el entorno software de ejecución de aplicaciones de RV elegido?
- Permiten el Hw y Sw gráfico de la plataforma de ejecución cumplir con los requerimientos de calidad de imagen establecidos?
- ¿Qué programas de modelado hay disponibles para la plataforma elegida? ¿Qué facilidades de modelado y prueba ofrecen?

Pero hay otros factores más sutiles, que es necesario analizar con precaución:

- ¿Voy a emplear una única plataforma para todas las tareas que el sistema de RV debe llevar a cabo o, por el contrario, voy a ir a una solución distribuida? En este último caso, ¿cómo van a conectarse las distintas plataformas entre sí?
- El sistema de desarrollo de aplicaciones de RV requiere el uso de un compilador? Si es así, ¿con qué compiladores, de los que corren sobre mi plataforma de desarrollo, es compatible?, ¿qué funcionalidades ofrece cada uno de ellos, especialmente en lo que se refiere a depuración?
- ¿Dispone mi software de desarrollo de aplicaciones de RV de drivers para los periféricos seleccionados? A menudo podemos encontrarnos con que el entorno de ejecución de aplicaciones de RV sólo admite unos determinados tipos de visiocasco o de dispositivo de localización.
- Si mi software de desarrollo de aplicaciones de RV no dispone de los drivers requeridos, ¿dispone al menos de las herramientas necesarias para desarrollar mi propio driver? Algunos sistemas de desarrollo de aplicaciones de RV tienen incluidas las herramientas necesarias para añadir nuevos drivers en caso necesario; en otros, es necesario comprar como opción los módulos precisos para desarrollarlos y en otros,

finalmente, no es posible utilizar más que los periféricos incluidos en la lista de dispositivos soportados.

- En caso de no utilizar un entorno de ejecución comercial, ¿los periféricos elegidos disponen de drivers para el sistema operativo específico que deseo utilizar? ¿Funcionan dichos drivers con el compilador elegido para el desarrollo de la aplicación?
- El programa de animación y modelado 3D elegido ¿proporciona un formato de salida que el sistema de desarrollo de aplicaciones de RV pueda entender? ¿Es este formato suficientemente optimizado para mi aplicación? Si no, ¿dispongo de la suficiente información para construirme un conversor? ¿Qué información tengo sobre los formatos internos del programa de animación, que pueden ser más útiles?

También es necesario evaluar de modo cuidadoso la capacidad del sistema *y de cada uno de sus componentes*, procurando alcanzar una solución homogénea:

- ¿Están bien compensadas las capacidades gráficas de la plataforma y del dispositivo de visualización? No tiene sentido, por ejemplo, emplear el dinero en aceleradores gráficos extremadamente potentes, si luego la pantalla utilizada no permite sacar partido de sus características. De la misma forma, comprar un sistema binocular de alta resolución puede ser un desperdicio si luego no voy a disponer de un acelerador gráfico de la suficiente calidad.
- ¿Va a utilizarse una velocidad de generación de las imágenes fija o variable? Si es fija, ¿cuál es la carga gráfica máxima del sistema, es decir, cuál es la máxima complejidad gráfica de una escena? Si no se limita la carga máxima en fase de diseño, habrá que recurrir a técnicas de control de complejidad, para poder mantener en todo momento la velocidad de generación de las imágenes especificada.
- ¿Sucede algo en mi aplicación si de vez en cuando se pierde una imagen, al no haber dado tiempo a generarla? En algunas aplicaciones, como muchas de ocio, no pasa absolutamente nada porque una imagen se pierda de vez en vez. En consecuencia, podemos permitirnos ciertas liberalidades con la carga gráfica máxima del sistema.
- ¿Están bien compensadas la capacidad de procesamiento geométrico y la de procesamiento ráster, para mi tipo concreto de aplicación?
- ¿La latencia máxima del sistema está dentro de los límites deseados? ¿Cuál es el tiempo de localización, de simulación y de síntesis de la escena, en el caso peor? ¿Cuál es la latencia que la red introduce, en caso de utilizar un sistema distribuido? ¿Está la suma de dichos tiempos dentro del límite marcado para la latencia?
- ¿Están bien compensadas las capacidades gráficas y sonoras del sistema? En muchas aplicaciones, como el diseño arquitectónico, el sonido puede jugar un papel menor, pero en otras muchas, no utilizar un sonido de la adecuada calidad puede dar completamente al traste con el objetivo impuesto. A la inversa, tampoco tiene sentido emplear el dinero en costosos dispositivos de audio, si éstos no añaden verdaderamente nada a la aplicación desde el punto de vista del "realismo" o del aspecto estético final.

Es posible que haya factores que sean fijos (por ejemplo, es posible que la plataforma tenga que ser una determinada), pero con respecto a los demás factores, mi consejo es que tome una decisión global. Seleccionar unos componentes antes que otros, a menos que esté muy seguro de lo que hace, puede darle problemas de compatibilidad insospechados.

Un consejo: no se fíe lo más mínimo de las especificaciones de los fabricantes. Los mercados avanzados son particularmente propicios a que cada cual dé a los conceptos el significado que le venga en gana. Siempre que pueda, pida una demostración de los productos, procurando efectuar comparaciones lo más objetivas posibles.

Quizá el ejemplo más paradigmático sea el referente a la velocidad de síntesis de una plataforma o de una tarjeta gráfica. Cuando alguien dice que su tarjeta gráfica es capaz de sintetizar 300.000 polígonos por segundo, en realidad no está diciendo absolutamente nada. Para empezar, se trata de una cifra que sólo nos habla de la velocidad de procesamiento geométrico, sin proporcionarnos ningún dato de la capacidad de rasterización de la tarjeta.

Pero, además, ¿en qué condiciones se ha hecho esa medida? ¿Se trata de polígonos con sombreado plano o de Gouraud? ¿Cuántas fuentes de luz se han utilizado para el sombreado? ¿Se ha utilizado un conjunto aleatorio de polígonos o el conjunto tenía una estructura optimizada? ¿La medida es con o sin eliminación de caras ocultas? ¿Con o sin selección? ¿Con o sin clipping? ¿Se ha realizado una transformación geométrica de coordenadas para los polígonos, o la posición de éstos era fija?

Dependiendo de la respuesta a estas preguntas, la velocidad de síntesis real puede ser inferior en un orden de magnitud (o incluso más) a la que el fabricante declara. Por eso la mejor manera de dilucidar la capacidad real de un sistema es por comparación: someta a los distintos sistemas a un mismo tipo de prueba, y compare los resultados. Si es posible, diseñe un pequeño test representativo de las situaciones que va a encontrar cuando ejecute la aplicación.

7.4. DISEÑO GRÁFICO DE LOS OBJETOS

Una vez integrado el sistema de RV y diseñada la estructura de nuestra aplicación, el diseño de los distintos objetos que compongan el mundo virtual va a seguir dos caminos paralelos. Por un lado, será necesario proceder con la labor de creación artística que implica el dotar a los objetos de su apariencia gráfica y sonora. Por otro lado, será necesario desarrollar el programa mediante el que se implementará el comportamiento de los objetos y la funcionalidad de nuestro mundo virtual.

7.4.1. Pasos del proceso

La creación de la apariencia gráfica y sonora de los objetos estará estrechamente ligada con el comportamiento que para los objetos se haya elegido. La figura 7.2 muestra un esquema de los pasos que componen el proceso de creación artística de los objetos.

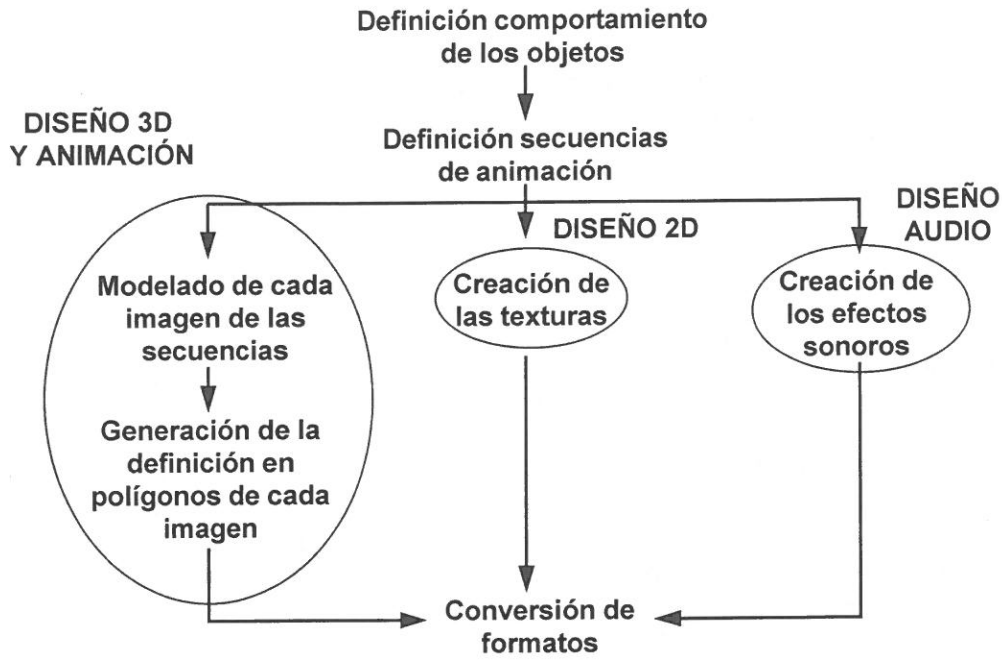

Figura 7.2. Creación artística de los objetos

El comportamiento de los objetos se traducirá en que, dependiendo de los eventos que tengan lugar durante el desarrollo de la aplicación, se ejecutarán una o más secuencias de animación para ellos (salvo que se trate de objetos de forma perfectamente fija).

Cada una de esas secuencias se compondrá de una serie de imágenes sucesivas que habrá que modelar. Normalmente, los programas de desarrollo de aplicaciones de RV suelen incorporar herramientas para el modelado de objetos 3D, pero las facilidades que ofrecen suelen ser bastante simples. También existen programas excelentes de modelado 3D, como AutoCAD, pero no resuelven el problema de la creación de animaciones.

Por eso, para modelar las imágenes (excepto si son muy sencillas), conviene emplear algún paquete especializado de animación, desde el que luego se exportan los datos al entorno de ejecución elegido.

Una vez modelada cada imagen de la secuencia, será necesario construir la definición correspondiente en polígonos. Los paquetes de animación comerciales suelen emplear, como ya hemos comentado anteriormente, formatos internos de representación de los objetos, a partir de los cuales pueden generarse representaciones poligonales con más o menos nivel de detalle.

El nivel de detalle que elijamos determinará el número total de polígonos del objeto y, en consecuencia, afectará a la capacidad del sistema: cuantos más polígonos empleemos para representar cada objeto, mayor es la carga que colocamos sobre el proceso de síntesis de las escenas. En caso de utilizar mecanismos de variación del nivel de detalle en nuestro algoritmo de síntesis, deberemos generar tantas representaciones poligonales de cada imagen como niveles de detalle maneje nuestra aplicación.

Es posible que utilicemos texturas en la representación de alguno de los objetos de nuestro sistema, o que alguno de los objetos se represente mediante un bitmap auto-orientable. En ese caso, será necesario diseñar los correspondientes bitmaps, bien utilizando un programa de diseño 2D, bien empleando las facilidades de diseño de texturas de nuestro paquete de animación 3D (si es que posee dicha funcionalidad). La mayoría de los paquetes de edición de bitmaps (como el que viene incluido con CorelDraw) poseen numerosas funciones avanzadas que nos serán extremadamente útiles a la hora de diseñar texturas. Es el caso, por ejemplo, de las funciones de filtrado, que nos ayudarán a reducir el problema del aliasing.

Asimismo, las secuencias de animación llevarán aparejados, posiblemente, determinados efectos sonoros: el ruido de un objeto al explotar, o el de unos pasos sobre el suelo. Dichos efectos deberán ser generados empleando alguno de los múltiples programas existentes en el mercado para todo tipo de plataformas.

Una vez generado el material gráfico y sonoro, será necesario convertirlo a los formatos que acepte el entorno de ejecución de aplicaciones de RV elegido. Las herramientas software utilizadas durante la fase de diseño artístico deberán disponer de las facilidades de exportación necesarias, o el programa de desarrollo de aplicaciones de RV disponer de opciones para importar los datos a partir de los formatos nativos de los programas de diseño utilizados.

7.4.2. Modelado de objetos 3-D

Una vez identificadas, a partir de los diagramas de animación, las distintas imágenes que hay que modelar para cada uno de los objetos de la aplicación, habrá que definir su geometría, utilizando alguna herramienta de diseño gráfico. Para la definición de la forma (geometría) de los objetos existen dos métodos fundamentales: constructivos y algorítmicos.

7.4.2.1. Métodos constructivos

Casi todos los programas de modelado 3D incorporan una serie de herramientas para la definición y edición de curvas y superficies primitivas: polígonos, esferas, rectángulos, cilindros, etc.

Utilizando dichas herramientas, pueden construirse los distintos componentes de un objeto: su imagen se diseña descomponiéndola en partes, y aproximando cada una mediante alguno de los objetos.

Para poder componer la imagen del objeto de esta manera, los paquetes de modelado incorporan otra serie de herramientas que permiten generar nuevos objetos a partir de estas

construcciones primitivas, mediante combinación de los elementos simples, transformaciones, mezclas, extrusiones o revolución de curvas 2D.

Otras técnicas más sofisticadas incluyen la definición de splines y NURBS (Non Uniform Rational Basis Splines), curvas y superficies en las que se definen una serie de puntos de control con los que luego puede manipularse la curva o superficie (como si se tratara de manillas), de forma que los puntos vecinos también se modifican, creándose una superficie uniforme.

También puede generarse la imagen de un objeto utilizando técnicas de digitalización de algún modelo real, de las que luego hablaremos.

7.4.2.2. Métodos algorítmicos

Los métodos algorítmicos permiten generar modelos tridimensionales de objetos sin necesidad de construirlos a partir de sus partes componentes. Básicamente, lo que se hace es emplear algún tipo de algoritmo de generación de carácter matemático, que da como resultado el modelo del objeto.

Este tipo de técnica esta limitada a aquellos tipos de objetos para los cuales existe un algoritmo de generación conocido. Entre los más populares están los algoritmos de generación de plantas, montañas y texturas mediante fractales, la generación de plantas y árboles mediante modelos de derivación gramatical, y los programas de deformación de sólidos y superficies planas (para simulación de pliegues en las ropas). Los métodos algorítmicos que podamos utilizar vendrán determinados por las opciones de que nuestro programa de animación 3D disponga.

7.4.3. Animación en 3-D

Como ya hemos mencionado anteriormente, una animación consta de una serie de representaciones gráficas distintas de un objeto, correspondientes a distintas instantáneas de una cierta actividad que dicho objeto desarrolla como parte de su comportamiento.

En el caso de objetos sencillos, y objetos compuestos a partir de objetos sencillos, podemos ir modelando manualmente cada una de las representaciones de la secuencia. Un ejemplo sería la animación correspondiente a un coche, donde tan sólo se mueven las ruedas: dado que la mayor parte del objeto no cambia de una a otra representación gráfica, podemos muy fácilmente editar los cambios, entre una representación y la siguiente, de manera manual.

Pero en el momento en que tratemos con objetos algo más complejos, como puede ser una figura humana, diseñar por separado cada representación gráfica de una secuencia puede ser una tarea bastante tediosa.

Por eso los programas de animación comerciales utilizados en el desarrollo de aplicaciones de RV suelen proporcionar herramientas para ayudar al diseñador a generar más fácilmente secuencias de modelos (animaciones). Utilizando dichas herramientas, el diseñador tan sólo tiene que modelar en detalle una o dos de las representaciones gráficas de la

secuencia. A partir de ahí, el paquete de animación puede generar de manera automática las restantes representaciones, bajo control del operador. Una vez terminado el trabajo de generación automática, lo único que hay que hacer, si se desea, es efectuar el retoque de las figuras resultantes.

Los métodos más empleados son el de **interpolación**, el de **cinemática inversa** y el de **simulación dinámica de sistemas de partículas**.

El método de **interpolación** permite generar las imágenes intermedias de una secuencia a partir de las representaciones gráficas inicial y final. Este método se aplica sobre todo a objetos no articulados, y es adecuado siempre y cuando la representación inicial y la final no difieran excesivamente, y no haya variaciones bruscas de velocidad en la secuencia.

Los métodos de **cinemática inversa** se aplican a la construcción de animaciones para objetos articulados, como animales o robots. El programa de animación permite al diseñador definir el "esqueleto" del objeto, alrededor del cual se modela la primera de las imágenes. El paquete de animación contiene una serie de reglas referentes al movimiento de los objetos articulados, mediante las cuales puede "mover" la figura del objeto. Lo único que tiene que hacer el diseñador (aparte de definir el esqueleto del objeto) es muestrear la secuencia resultante, para quedarse con el número de representaciones gráficas distintas que desee. Después, puede hacerse un retoque manual de cada una de las imágenes de la secuencia, con el fin de mejorar el aspecto.

La **simulación dinámica de sistemas de partículas** se emplea para modelar secuencias de animación de objetos que puedan representarse mediante un gran número de partículas de pequeño tamaño. Se puede utilizar esta técnica, por ejemplo, para modelar explosiones de objetos, para modelar columnas de humo, nubes, fuego, ... El diseñador no tiene más que inicializar los parámetros del sistema de partículas, y pedir al paquete de animación que genere de manera automática un cierto número de imágenes de la secuencia. Dichas imágenes pueden luego, asimismo, retocarse de manera manual. Después, dichas representaciones gráficas se almacenan para su uso posterior por la aplicación. Lo normal es que este tipo de representaciones tridimensionales se almacene en forma de bitmap, para su uso como bitmap auto-orientable, y no como objeto tridimensional.

Algunos paquetes de animación comerciales disponen de algunos otros tipos de herramientas para producir animaciones: herramientas de deformación de objetos, herramientas para generar perturbaciones similares a las olas en la superficie de los objetos, etc..., pero dichas herramientas tienen una utilidad más limitada.

7.4.4. Generación de la base de datos gráfica

Una vez diseñados los objetos componentes de cada una de las secuencias de animación, es necesario generar las correspondientes representaciones poligonales. En el caso de objetos sencillos de caras planas, el proceso es inmediato; pero para objetos que posean superficies curvas se hace necesario proceder a muestrear dichas superficies con el nivel de detalle requerido.

La resolución (nivel de detalle) que elijamos debe ser adecuada al hardware gráfico utilizado. Cuantos más objetos puedan coexistir en una escena, y más polígonos tenga la representación gráfica de los mismos, mayor será la capacidad de proceso necesaria para sintetizar una escena. Lo más probable es que, durante la fase de integración final de la aplicación, nos veamos a menudo forzados a volver atrás y variar el nivel de detalle de los objetos menos importantes, con el fin de disminuir la carga de procesamiento gráfico.

Después de muestreada la figura, el conjunto poligonal así obtenido deberá exportarse a un formato de fichero que el sistema de desarrollo de aplicaciones de RV elegido reconozca.

7.4.5. Programas de modelado y animación comerciales

Los programas de animación comerciales se utilizan ampliamente para animaciones profesionales dirigidas al cine o la TV. En realidad, es éste (y no el de los sistemas de tiempo real) el principal mercado para este tipo de herramientas (por lo menos hasta ahora).

En consecuencia, las herramientas de animación comerciales suelen incorporar diversas facilidades que son completamente inútiles para el desarrollo de aplicaciones de RV[1].

Los paquetes de animación comerciales incorporan, por ejemplo, una serie de herramientas y técnicas pensadas específicamente para mejorar la calidad de las imágenes: técnicas de radiosidad, trazado de haces, reflexiones, transparencias, efectos, etc... Estas técnicas permiten conseguir una extraordinaria dosis de realismo, pero no pueden emplearse en sistemas en tiempo real como los de RV, por el alto tiempo de computación requerido.

También incorporan estos programas herramientas para el manejo y definición de escenas, incluyendo facilidades como movimientos de las cámaras, movimientos de las luces, traslaciones de los objetos, etc... Todas estas facilidades se proporcionan porque el propio paquete con el que se diseñan las figuras es el que luego se utilizará para generar las imágenes de las escenas empleadas en el mundo publicitario. Sin embargo, en nuestro caso, la generación de las escenas se realiza en tiempo real utilizando las facilidades que el entorno de ejecución proporcione, por lo que todas estas funcionalidades del paquete de animación nos son inútiles.

Así que, a la hora de evaluar un paquete comercial de modelado y animación 3D, lo más importante es su capacidad de modelado, de animación *de objetos* y los formatos de entrada y salida de datos que admita; y lo menos importante son los módulos de síntesis de escenas, las capacidades de iluminación y sombreados, las técnicas de trazado de haces y radiosidad, y las facilidades de animación *de escenas* (excepto como herramienta interactiva para hacer pruebas).

---

1 Otra consecuencia del origen (tan alejado de los sistemas de tiempo real) de estas herramientas es que todas ellas permiten exportar a numerosos formatos, pero ninguna suele disponer de formatos optimizados para el tratamiento de los datos en tiempo real.

Hay que evaluar muy cuidadosamente la inversión que supone adquirir uno de estos paquetes. Si la plataforma utilizada para ejecutar la aplicación de RV no dispone de unas capacidades gráficas elevadas, puede no tener mucho sentido adquirir un complejo programa de animación, porque no vamos a sacar partido de las posibilidades gráficas. En esta circunstancias, puede ser más adecuado emplear las herramientas simples de modelado incorporadas en los propios sistemas de desarrollo de aplicaciones de RV.

Casi todos (por no decir todos) los paquetes profesionales existentes para el desarrollo de animaciones corren sobre estaciones Silicon Graphics, aunque existen algunos programas para PC de gran calidad. Todos suelen ir de 1 a más de 5 millones de pts, dependiendo de las opciones.

Los más populares de entre estos paquetes son:

- **Wavefront**: Incorpora facilidades para el modelado de objetos 3D (incluyendo NURBS), definición de materiales, animación de caracteres, simulación dinámica y deformaciones. Entre las compañías del mercado de aplicaciones gráficas de tiempo real que utilizan Wavefront destaca la compañía de desarrollo de juegos Spectrum Holobyte.
- **Alias**: Incorpora facilidades de modelado de objetos 3D similares a las de Wavefront. El módulo de animación de objetos dispone de herramientas de cinemática inversa, interpolación y deformaciones. Alias es el paquete elegido por Nintendo para sus desarrollos.
- **SoftImage**: La compañía canadiense SoftImage ha desarrollado y comercializa el paquete de modelado y animación Creative Environment, elegido por la compañía Sega para el desarrollo de alguno de sus videojuegos, como Virtua Racing. Dispone de un modelador de objetos 3D (que no tiene NURBS, aunque sí herramientas de creación y manipulación de splines), un módulo de definición de materiales para determinar propiedades de los objetos como texturas o transparencias, y un módulo de animación de caracteres mediante cinemática inversa y simulación dinámica. También incorpora una serie de herramientas auxiliares para la deformación de objetos, creación de efectos atmosféricos y conversión de ficheros, así como el paquete AMAP desarrollado por el CIRAD francés, para la creación de plantas en 3D.

  Creative Environment tiene un kit de desarrollo que proporciona el código fuente necesario para escribir aplicaciones que interactúen con el programa, incluyendo información sobre la estructura interna de la base de datos.

Mencionamos únicamente las características que mayor importancia tienen para aplicaciones interactivas de tiempo real. Todos estos paquetes disponen, por supuesto, de muchas otras facilidades adecuadas a la generación de escenas de alta calidad.

Entre los otros programas comerciales existentes (aunque menos extendidos) para estaciones de la gama alta, podemos mencionar Vertigo, de la empresa Vertigo Technologies, que utiliza el paquete de síntesis de imágenes RenderMan, de Pixar; ModelGen, de Software Systems; 3DGO, de ElectroGIG; y VisLab, de Engineering Animation, que también emplea RenderMan.

Existen, asimismo, numerosos paquetes de animación para PC, que suelen ofrecer menos funcionalidades pero, a cambio, tienen un precio más asequible. Entre ellos podemos destacar los programas Playmation o aniMeister, los dos para entorno Windows, aunque el mejor de

todos ellos sea, posiblemente, 3D Studio, de Autodesk. 3D Studio incluye un paquete de modelado de objetos 3D (sin NURBS), edición de materiales y efectos especiales. También incluye facilidades de animación, aunque más limitadas.

7.4.6. Técnicas y programas de digitalización

Una de las posibilidades existentes a la hora de diseñar figuras tridimensionales consiste en emplear un modelo real y digitalizarlo, capturando los datos en un soporte informático, para su posterior edición. Para realizar la digitalización de las figuras puede emplearse cualquier tipo de dispositivo de localización (electromagnético, mecánico, ultrasónico, ...), aunque la tecnología de ultrasonidos y la mecánica presentan la ventaja, sobre los dispositivos electromagnéticos, de permitir digitalizar materiales férricos.

Existen compañías, como por ejemplo las americanas Viewpoint y Syndesis Corporation, que venden modelos tridimensionales digitalizados, para su utilización en cualquier animación o programa de realidad virtual o de modelado. Dichos modelos pueden encargarse en cualquiera de los formatos más populares, generalmente en el del programa software con el que vayan a emplearse: DXF, Wavefront, 3D Studio, SoftImage, AutoCAD, ...

Hay programas comerciales para la digitalización de modelos tridimensionales que pueden adquirirse con el fin de realizar uno mismo la digitalización de los modelos. Uno de tales programas es Kheops, de la empresa belga Little Big One, que corre sobre Silicon Graphics y PC, y emplea un sensor de posición Polhemus para la captura de los puntos. Una serie de utilidades de edición permite efectuar los retoques en el modelo digital una vez capturado.

La empresa americana SAC comercializa el digitalizador sónico GP-8-3D. Este dispositivo emplea ondas sonoras para determinar la distancia entre el punto que se desea digitalizar y cuatro micrófonos situados en posiciones fijas. Al accionar el conmutador situado en la cabeza de medida, ésta emite un impulso ultrasónico y comienza a medirse el tiempo que el impulso tarda en llegar a cada uno de los micrófonos. Con los cuatro valores de retardo, la unidad de control calcula la posición relativa del punto objeto de medida. El tamaño máximo del objeto por digitalizar es de 2.5 m de arista, y la resolución es de 0.1 mm. La unidad de control se conecta al ordenador a través de un puerto serie o paralelo.

La también americana Immersion Corporation comercializa un digitalizador mecánico, que puede emplearse con objetos de pequeño tamaño (hasta unos 30 o 40 cm de arista) y se conecta al ordenador a través del puerto serie.

Una vez capturados los datos, lo normal es importarlos desde el programa de animación utilizado, con el fin de efectuar retoques, definir propiedades de los materiales, o realizar animaciones partiendo de dicha figura base.

7.4.7. Formatos gráficos

Entre los principales formatos gráficos para aplicaciones de realidad virtual, podemos mencionar:

- **DXF** (Drawing Interchange Format): Formato de Autodesk, utilizado en su programa AutoCAD. Es un formato muy popular, además de por la gran utilización de AutoCAD, por el hecho de que es un formato ASCII puro bastante sencillo, lo que facilita la realización de herramientas de conversión. Casi todos los programas de modelado o animación lo admiten, así como casi todos los programas de desarrollo de aplicaciones de RV, por lo que está muy extendido, tanto dentro del mundo Unix como en el del PC.
- **IGES** (Initial Graphics Exchange Specification). Es un estándar ANSI para la definición de objetos tridimensionales, ampliamente empleado en la industria manufacturera, especialmente en EEUU. Es muy antiguo, y va a ser sustituido por el estándar STEP, por lo que no se emplea excesivamente en aplicaciones de RV.
- **STEP**: Estándar ISO-10303, con el que se pretende sustituir a IGES. Pensado para la industria manufacturera, sus características también se adaptan a la definición de los modelos tridimensionales empleados en aplicaciones de RV. Sin embargo, el estándar es aún demasiado reciente y su uso no está todavía extendido.

Además de estos formatos, más o menos estándar, cada uno de los paquetes de animación comerciales emplea sus propios formatos de definición de objetos 3D, siendo bastante común el intercambio de datos en formato, por ejemplo, Wavefront o SoftImage.

Hay que destacar que ninguno de los formatos mencionados está optimizado para aplicaciones de tiempo real. Las aplicaciones de RV se han empezado a generalizar muy recientemente, y no existen todavía formatos estándar que traten de explotar la estructura de los objetos con el fin de acelerar la velocidad de síntesis de las imágenes. Tampoco existen, por el momento, formatos estándar suficientemente extendidos para la definición de la estructura de las escenas.

7.5. ENTORNOS DE DESARROLLO

7.5.1. Funciones

El proceso de creación de los objetos que formen parte de nuestra aplicación de RV implica no sólo definir las representaciones gráficas de los mismos, sino también desarrollar el programa en el que se implementará el comportamiento de cada objeto en particular, y del mundo virtual como un todo.

La definición e implementación del comportamiento de los objetos se realizará en algún lenguaje de programación de alto nivel, haciendo uso de las facilidades ofrecidas por el programa de desarrollo de aplicaciones de RV elegido.

El comportamiento de los objetos puede ser controlado por el usuario, a través de algún dispositivo de entrada, o por el sistema. En la literatura sobre RV, este último tipo de objeto se denomina algunas veces "agentes autónomos".

A la hora de la creación de los objetos, el entorno de desarrollo permitirá definir para cada uno las correspondientes funciones que implementen el comportamiento, asociarle las

representaciones gráficas, animaciones y secuencias de sonido pertinentes, e importar los correspondientes ficheros.

Una vez implementado cada uno de los objetos individuales que intervengan en la aplicación, el entorno de desarrollo nos permitirá definir la estructura global de nuestro mundo virtual y cada una de las escenas componentes.

De cara a la definición de escenas, los entornos de desarrollo permiten definir, entre otras cosas:

- Qué objetos van a formar parte de cada escena y cuál es su posición inicial.
- Posición (y comportamiento) de las luces.
- Cuáles son los límites de la escena (paredes y suelo).
- Posición de las cámaras.
- Lugares o condiciones de paso a otras escenas (portales).

Asimismo, el entorno de desarrollo nos permitirá asignar cada uno de los dispositivos periféricos a uno o más objetos del mundo virtual. Como ejemplo, lo normal será que se establezca un enlace entre el dispositivo de localización utilizado para determinar la posición del usuario y la cámara situada dentro de la escena; también podría establecerse un enlace entre alguno de los objetos móviles de la escena y un dispositivo tipo joystick.

Éstas son las funcionalidades básicas que cualquier sistema de desarrollo de aplicaciones de RV suele ofrecer, pero algunos sistemas pueden presentar funcionalidades más elaboradas, como la definición de estructuras jerárquicas entre los objetos, o herramientas de definición de la interfaz de usuario para el caso de aplicaciones de sobremesa.

Normalmente, la forma en que estas funcionalidades se ofrecerán al programador será mediante un conjunto de librerías, cuyas funciones se invocarán de acuerdo con las necesidades de nuestra aplicación.

A continuación vamos a describir brevemente los entornos de desarrollo de las aplicaciones de RV más populares. Todos los entornos de desarrollo mencionados en este apartado disponen de soporte para diversos periféricos de RV y un modelador sencillo de objetos que permite definir tanto su geometría 3-D como su comportamiento (aunque con bastantes limitaciones). Algunos disponen, además, de facilidades para la incorporación de texturas y de la posibilidad de conectar varias máquinas en red.

7.5.2. Worldtoolkit

7.5.2.1. Descripción del producto

La empresa americana Sense8 es la compañía líder en el mercado de los sistemas de desarrollo de aplicaciones de realidad virtual. Sense8 comercializa un producto denominado WorldToolKit, que es una librería de funciones C que permite desarrollar aplicaciones de RV sobre distintas plataformas.

WorldToolKit permite desarrollar aplicaciones de alta resolución (dependiendo, obviamente, de la plataforma), con texturas de 24 bits por color. Incluye rutinas de anti-aliasing, construcción geométrica y definición de movimientos de los objetos, texturas y sombreado. Incluye también rutinas de interfaz con los periféricos de RV más populares (visiocasco de Virtual Research, gafas CrystalEyes de StereoGraphics, etc...) y herramientas para la construcción de drivers para nuevos periféricos.

Entre las opciones de expansión disponibles, una de ellas permite la construcción de aplicaciones multiusuario en red y otra permite conectar dos sistemas en modo maestro-esclavo, obteniéndose un rendimiento de hasta un 70% más que en la versión monomáquina.

Permite importar definiciones gráficas de objetos diseñados con programas de animación comerciales como 3D-Studio o Wavefront.

WorldToolKit es comercializado en España por la empresa Tecnología Virtual.

7.5.2.2. Versiones

Las versiones existentes de WorldToolKit son:

- WorldToolKit for Windows: versión de bajo coste que corre sobre Windows 3.1, realizando todas las labores de síntesis de la imagen por métodos exclusivamente software. Soporta los siguientes compiladores: Borland C++ 4.0, Watcom C++ 9.5, Microsoft Visual C++ y Symantec C++. No dispone de drivers para dispositivos inmersivos.
- WorldToolKit 860: corre sobre cualquier 486 o Pentium equipado con la tarjeta de síntesis de imágenes Fire 860 de Spea, DOS versión 3.0 o superior y el DOS-extender de Pharlap. Las capacidades gráficas que ofrece son las de la tarjeta Fire: resolución de hasta 800x600, texturas de hasta 256x256, sombreado de Gouraud. Utilizando dos tarjetas gráficas, pueden construirse aplicaciones estereoscópicas.
- WorldToolKit SGI: corre sobre cualquier estación Silicon Graphics con sistema operativo IRIX 4.0.5 o superior, librería gráfica GL versión 4.0, y API Performer 1.x. Las capacidades gráficas dependen de la estación concreta escogida, de entre la gama de SGI.
- WorldToolKit SUN: corre sobre cualquier estación SPARC2, SPARC10 o SPARC20 equipada con el acelerador gráfico ZX. Requiere sistema operativo Solaris 2.3, librería gráfica XGL 3.0 y compilador ANSI C. Utilizando el sistema operativo SunOS 4.1.3 puede emplearse un acelerador gráfico de la serie Freedom de Evans & Sutherland. Esta versión de WorldToolKit no soporta texturas.
- WorldToolKit DEC: Es la versión de WorldToolKit para estaciones Alpha, de Digital. Requiere sistema operativo OSF 1.3, y acelerador gráfico Denali versión 2.10 o superior, de Kubota Pacific. Dado que Kubota ha decidido abandonar la línea de aceleradores gráficos para Alpha, ésta es una versión en vías de extinción.

7.5.2.3. Periféricos soportados

La lista de periféricos soportados incluye:

Aceleradores gráficos para PC	• Fire 860, de Spea (sólo para WTK 860) • Viper, de Diamond Computer (sólo para WTK for Windows, sólo admite sombreado plano)
Gafas estereoscópicas	• CrystalEyes, de StereoGraphics
Visiocascos	• Eyegen3, de Virtual Research • VR4, de Virtual research
Sistemas binoculares	• Boom 3C, de FakeSpace Labs (sólo para WTK SGI, WTK SUN y WTK DEC)
Ratones 3D	• Logitech (para WTK for Windows)
Controladores 3D	• Spaceball 2003, de Spaceball Technologies (todas las versiones excepto WTK for Windows)
Sensores de posición	• FASTRAK, de Polhemus • ISOTRAK2, de Polhemus • InsideTRAK, de Polhemus (sólo WTK 860)
Electroguantes	• DataGlove Model 2, de VPL • CyberGlove, de Virtual Technologies
Tarjeta de sonido 3D	• Beachtron, de Crystal River Engineering (sólo WTK 860)

7.5.3. Superscape

7.5.3.1. Descripción del producto

Superscape, empresa británica que antes se denominaba Dimension International, es una de las compañías líderes en el mercado de la realidad virtual de sobremesa, y una de las dos compañías europeas presentes en el campo de los sistemas de desarrollo de aplicaciones de RV.

Superscape fue fundada en 1983 como una compañía de desarrollo de juegos, y cuatro años más tarde desarrollaron Freescape, un software de generación de escenas tridimensionales en tiempo real, con el que se introdujeron en el mercado de la realidad virtual. Un año después, en 1988, comenzaron el diseño de un sistema completo de desarrollo de aplicaciones de RV para PC, llamado Superscape, que saldría al mercado en 1991.

Desde su lanzamiento, Superscape ha ido incorporando una serie de mejoras en las sucesivas versiones. Inicialmente pensado para la construcción de aplicaciones de RV de sobremesa, fue posteriormente ampliado para permitir el desarrollo de sistemas

inmersivos y proyectivos. El producto, que obtuvo el premio otorgado por la revista especializada CyberEdge, tuvo un éxito considerable, lo que hizo que la compañía cambiara de nombre.

La configuración mínima recomendada para Superscape es un 486 a 33 MHz, con 8 Mbytes de RAM, 100 MBytes de disco duro, monitor color de 14 pulgadas, MSDOS 5.0 o superior, ratón, joystick, tarjeta SVGA con 512K de RAM y disco de 3.5 pulgadas.

7.5.3.2. Opciones

El producto básico es el Superscape VRT3, que es un sistema interactivo de creación y edición de mundos virtuales. El VRT3 incluye:

- Editor geométrico, con el que se pueden definir objetos en tres dimensiones, incluyendo texturas.
- Editor de escenas, con el que se puede definir la estructura de las escenas e indicar los objetos componentes.
- Editor de pantallas y recuadros de diálogo, que permite definir la interfaz de usuario.
- Módulo de control de objetos, responsable de llevar a cabo la simulación.
- Sistema de visualización en tiempo real (denominado Visualizer), responsable de realizar la representación gráfica de las escenas.
- Componentes auxiliares, que incluyen un editor de sonido, un editor de teclado, un módulo de conversión de ficheros DXF, una librería de modelos geométricos y otra librería de aplicaciones ejemplo.

La programación de las aplicaciones se realiza mediante el Superscape Control Language, un lenguaje de programación similar al C.

El sistema básico es monousuario, pero la opción Superscape Networks, que puede adquirirse aparte como extensión de Superscape VRT3, permite la creación de mundos virtuales en red con un máximo de 8 usuarios concurrentes sobre una red Ethernet. El driver de red utilizado es Clarkson Packet Drivers, recomendándose las tarjetas 3COM 3C503 y 3C503/16.

Aquellos desarrolladores que deseen construir sus propios drivers para Superscape, o escribir una interfaz para Superscape con otros programas de aplicación, pueden adquirir como opción suplementaria el Superscape Developers Kit, que requiere el uso del compilador C de Watcom, versión 8.0 o superior.

Para poder distribuir a los usuarios finales varias copias de una misma aplicación, una vez desarrollada, es necesario adquirir un sistema de run-time por cada copia. El sistema de run-time incluye sólo los módulos necesarios para la visualización, sin ningún tipo de facilidad de edición.

La comercialización en España de Superscape corre a cargo de la empresa RTZ Technologies.

7.5.3.3. Periféricos soportados

La lista de periféricos soportados incluye:

Aceleradores gráficos para PC	• Tarjetas con interfaz TIGA, como las tarjetas Harlequin
Ratones 3D	• Logitech
Controladores 3D	• Spaceball, de Spaceball Technologies
Sensores de posición	• FASTRAK, de Polhemus
	• A Flock of Birds, de Ascension Technologies

7.5.4. dVISE

7.5.4.1. Descripción del producto

La empresa británica Division es, junto con Superscape, la única compañía europea presente en el campo de los sistemas de desarrollo de aplicaciones de realidad virtual.

Division suministra tanto hardware como software, así como sistemas completos. En el campo del software, Division comercializa dVISE, un sistema de desarrollo de aplicaciones de realidad virtual, así como el correspondiente entorno de ejecución. dVISE está preparado para funcionar con dispositivos de visualización estereoscópica y sonido tridimensional, así como dispositivos de localización y un amplio rango de periféricos. Dispone de módulos de conversión para poder importar objetos generados con la mayoría de los principales paquetes comerciales de animación 3D: AutoCAD, ProEngineer, MicroStation, Alias, MultiGen, Wavefront, 3D-Studio, etc...

El entorno de desarrollo proporciona dos modos diferentes de crear o editar un mundo virtual. En el modo más tradicional, se emplea una interfaz de ventanas X-Windows para modificar los atributos de los objetos mediante el teclado y el ratón. El otro modo es más llamativo: el desarrollador de aplicaciones puede introducirse en el propio mundo virtual, utilizando los periféricos adecuados, y modificar los objetos manualmente.

El entorno de ejecución consta de dos módulos básicos. El primero es el sistema de run-time, dVS, que se ejecuta en paralelo (aprovechando las posibilidades del sistema operativo Unix) con el código de la aplicación, y proporciona a ésta todos los servicios genéricos necesarios para su ejecución. El segundo módulo es el VCToolkit, una librería de funciones C que proporciona una interfaz de alto nivel para el desarrollo de aplicaciones.

7.5.4.2. Versiones existentes

dVISE corre sobre las estaciones gráficas PROVISION 100 de la propia Division (que son PCs EISA con sistema operativo Unix System V) y sobre estaciones Silicon Graphics.

7.5.4.3. Periféricos soportados

La lista de periféricos soportados incluye:

Aceleradores gráficos para PC	• PC dVIEW, de Division
Sistemas binoculares	• Boom, de FakeSpace Labs
Ratones 3D	• Logitech
Controladores 3D	• Spaceball, de Spaceball Technologies
Sensores de posición	• FASTRAK, de Polhemus
	• 3Space Tracker, de Polhemus
	• ISOTRAK, de Polhemus
	• A Flock of Birds, de Ascension Technologies
Electroguantes	• CyberGlove, de Virtual Technologies
Dispositivos de sonido 3D	• Beachtron, Convolvotron y Acoustetron, de Crystal River Engineering.

7.5.5. Otros sistemas de desarrollo

7.5.5.1. VREAM

VREAM es un programa de bajo coste para el desarrollo de aplicaciones de RV para PC, creado por la empresa del mismo nombre. Permite crear aplicaciones de sobremesa, utilizando el ratón y el teclado como dispositivos de control, o aplicaciones inmersivas que utilicen periféricos de realidad virtual como los visiocascos EYEGEN3 de Virtual Research, el ratón 3D de Logitech o los dispositivos de localización de Ascension Technologies. La configuración mínima recomendada es un 486DX a 33 MHz, equipado con tarjeta VGA estándar, 4 MBytes de memoria y 10 MBytes de disco duro.

VREAM permite programar las aplicaciones sin necesidad de usar ningún lenguaje de programación, aunque dispone de un lenguaje de scripting por si se desea definir el mundo virtual mediante un fichero de texto estándar. Al no requerir programación, se elimina la necesidad de disponer de otros paquetes software como compiladores o DOS extenders.

El sistema de desarrollo completo incluye:

- Editor de mundos tridimensionales.
- Intérprete de ficheros script.
- Módulo de run-time.
- Conjunto de drivers para dispositivos de sobremesa e inmersivos.
- Aplicaciones ejemplo.

Con el editor de mundos tridimensionales se modelan los objetos que componen la escena y a continuación se definen, haciendo uso de la interfaz gráfica, sus atributos (peso, movimientos, secuencias sonoras o elasticidad) y su comportamiento (expresado como una

serie de reglas del tipo condición/respuesta). Se pueden asignar texturas a los objetos, definir jerarquías de objetos (objetos compuestos) para las traslaciones y rotaciones, y fijar parámetros globales de la escena como la gravedad o las fuentes de luz. Se pueden importar objetos en formato DXF y bitmaps en formato PCX.

El editor de mundos genera como salida un fichero de texto, en el lenguaje de scripting, que sirve de entrada al módulo de run-time. Este módulo es el que va a permitir ejecutar la aplicación, pudiendo el usuario interactuar con los objetos del mundo virtual así creado, cogiéndolos o desplazándolos. Es posible, incluso, cambiar la escena en tiempo real, añadiendo o eliminando objetos mientras se está ejecutando la aplicación.

VREAM no permite desarrollar aplicaciones distribuidas. En España lo comercializa la empresa Tecnología Virtual.

La lista de periféricos soportados incluye:

Gafas estereoscópicas	• CrystalEyes, de StereoGraphics
Visiocascos	• Eyegen3, de Virtual Research
	• Tier1, de VRontier Worlds
	• MRG 2, de Liquid Image Corp.
Ratones 3D	• Logitech
Sensores de posición	• A Flock of Birds, de Ascension Technologies

7.5.5.2. Cyberspace Developer Kit

Autodesk ha desarrollado el Cyberspace Developer Kit (CDK), un conjunto de librerías de clases C++ y rutinas relacionadas para la construcción de aplicaciones de RV. Las clases de CDK permiten modelar la geometría y el comportamiento de los objetos tridimensionales, así como utilizar diferentes aceleradores gráficos, dispositivos de visualización y dispositivos de entrada. La geometría de los objetos se puede importar desde AutoCAD y 3D-Studio.

Posee herramientas básicas de definición de comportamiento, mediante las que es posible definir características básicas de los objetos y del mundo virtual, como la masa, fricción, gravedad, velocidad o volumen. El módulo de simulación permite detectar eventos como colisiones o intersecciones. CDK no tiene soporte de texturas.

7.5.6. El problema del comportamiento

En palabras de Steve Bryson, "el propósito de la inmersión es hacer que el usuario se sienta parte integrante del mundo virtual, interactuando con los objetos de éste igual que lo haría con los reales". Después de los años de desarrollo que lleva la RV, podemos decir que la primera parte se ha conseguido, pero la segunda dista mucho de estar resuelta.

Lo más comprendido del proceso de desarrollo de una aplicación de RV es la fase de modelado geométrico, porque ya viene haciéndose uso hace mucho tiempo de estas técnicas en los programas de CAD. En cuanto a los procedimientos de animación, los paquetes

comerciales, de gran uso en la industria del cine y de la TV, hace mucho que emplean técnicas bastante maduras.

Pero el tema del comportamiento no está, ni mucho menos, resuelto. Los entornos de desarrollo de aplicaciones de RV existentes proporcionan una serie de facilidades para la definición del comportamiento de los objetos. Empleando dichas facilidades, se puede dotar a los objetos de determinados comportamientos básicos como:

- Moverse, rotar.
- Ser empujados, cogidos o desplazados.
- Movimiento de las partes, en un objeto articulado.
- Cambios de escala.
- Cambios de apariencia (color, etc...).
- Animación (cambio de la representación gráfica).

Algunos entornos, como el CDK de Autodesk, incluyen algunas facilidades algo más elaboradas para la gestión de leyes físicas como la gravedad o la inercia. Sin embargo, si se desea un comportamiento más complejo, es necesario programarlo a nivel de aplicación.

En la actualidad, muchas aplicaciones de RV se emplean como herramientas de modelado (en arquitectura, urbanismo, decoración, turismo) o de prototipado para la industria de fabricación de vehículos y la industria aeronáutica. En este tipo de entorno, los requerimientos de simulación de comportamiento no son muy grandes, lo que quizá haya influido en los escasos avances producidos en el terreno de definición del comportamiento de los objetos. Pero la previsible explosión en los próximos años de los simuladores de RV, de las aplicaciones de ciberespacio y de los sistemas de ocio traerá, posiblemente, un cambio en el panorama, con el desarrollo de herramientas más sofisticadas para la definición de la lógica interna del mundo virtual. Hablaremos de ello en el próximo capítulo.

8
El futuro de la realidad virtual

El mercado de la realidad virtual presenta un potencial de crecimiento como pocos. O mejor deberíamos decir que quien presenta un potencial de crecimiento extraordinario es el mercado futuro en que la realidad virtual va a desembocar durante los próximos tres años. Porque en estos momentos se está produciendo una convergencia de mercados cuyo resultado nadie se atreve a predecir.

El mercado de los juegos, de las estaciones gráficas, de la multimedia, de los servicios on-line, de los programas de animación, de la realidad virtual... todos ellos están convergiendo a un ritmo cada vez más frenético, con juegos de alianzas en las que todo aquél que es alguien participa.

Desde Microsoft a Sega, desde Silicon Graphics a LucasFilm, todo el mundo (incluyendo a Compuserve, Philips, Nintendo, IBM, Time Warner, Matsushita, Panasonic, Sears, Atari, Evans&Sutherland, Sun, Namco, Virtuality, QuantumLink, Division, Kubota, Electronic Arts, Spectrum Holobyte, Apple, Fujitsu y unas cuantas decenas más de compañías) está tomando posiciones ante la avalancha que se avecina. Y esas posiciones por las que los distintos actores luchan de manera denodada lindan, todas ellas, con la realidad virtual.

Porque, a diferencia de otras tecnologías alrededor de las cuales tanta polvareda se levantó en su día (¿se acuerda usted de la inteligencia artificial o de los ordenadores de la quinta generación?), la realidad virtual (o cualquiera que sea el mercado que al final resulte de la mezcolanza) es un mercado que, hace ya mucho tiempo, que está proporcionando dinero a las compañías especializadas que han venido operando en él.

La tecnología de la realidad virtual va a cumplir ya treinta años. No es una nueva técnica para la que se anden buscando denodadamente aplicaciones. La excitación alrededor de la realidad virtual proviene del hecho de que la tecnología está alcanzando los costes necesarios para introducir las aplicaciones en el mercado de gran consumo. Y el tamaño de ese mercado puede ser inmenso.

Actualmente, las aplicaciones de la realidad virtual son múltiples, tanto en el campo de las aplicaciones profesionales como en el de las aplicaciones de ocio. Desde los simuladores

de vuelo hasta las aplicaciones de prototipado para la industria de la construcción de automóviles, los sistemas de RV están ya en prácticamente todas partes. En el campo del ocio, los juegos inmersivos son ya casi familiares, como pronto lo serán los juegos interactivos de plataforma.

En este capítulo, vamos a tratar de atisbar los caminos por los que, previsiblemente, los sistemas de RV discurrirán durante los próximos años. Y para entender dónde estamos situados, vamos a volver la vista atrás por unos instantes para repasar lo que ha sido la evolución de los sistemas de RV hasta la fecha.

8.1. UN POCO DE HISTORIA

El origen de la realidad virtual se sitúa, cómo no, en el campo de la investigación militar, hace aproximadamente unos veinticinco años. En aquella época (finales de los sesenta) el Departamento de Defensa americano estaba muy interesado en dos tipos de tecnología relacionados. De un lado, la tecnología de simulación de vuelo, porque los simuladores permitían entrenar a los pilotos sin riesgo para ellos ni para los aviones; además, la simulación permitía controlar el entrenamiento, fijando las condiciones ambientales y los escenarios específicos a voluntad. De otro lado, también estaba interesado en la investigación sobre dispositivos de control especializados para la conducción de aeronaves, especialmente cascos de visualización que permitieran combinar información gráfica e imágenes reales.

Durante los últimos años de la década de los sesenta y toda la década de los setenta, dichos objetivos iniciales darían lugar a una serie de proyectos en distintos centros de investigación de los Estados Unidos, como resultado de los cuales se fueron poco a poco sentando las bases de lo que la realidad virtual es hoy en día. Y no sólo la realidad virtual.

En primer lugar, el desarrollo de los simuladores de vuelo requería de la posibilidad de generar gráficos 3-D mediante ordenador, un campo éste prácticamente inexplorado hasta el momento. Como consecuencia, durante aquellos años se desarrollarían una serie de técnicas, algoritmos, herramientas y equipos pensados para el tratamiento de dicho tipo de información gráfica. Fue fundamentalmente en la Universidad de Utah donde, a partir de 1974, se desarrollaría la mayor parte del cuerpo de doctrina de la síntesis de imágenes. Allí, Ed Catmull inventó varios de los principales conceptos básicos de la síntesis de imágenes: la técnica del z-buffering, la técnica de alfa-blending y la técnica de texturado.

Pero la investigación sobre algoritmos de generación de imágenes 3-D no se limitó al objetivo inicial, sino que muy pronto se desgajaron dos distintas ramas de investigación: de un lado, la investigación destinada a los sistemas gráficos de tiempo real (como los simuladores de vuelo), donde el objetivo fundamental de los algoritmos y equipos es generar imágenes a gran velocidad con el fin de crear en el usuario la sensación de visión continua; de otro lado, numerosos investigadores comenzaron a definir algoritmos y herramientas para la obtención de imágenes sintéticas cada vez más realistas, eliminando el requisito de generación en tiempo real.

La primera de estas dos escuelas creó todo el cuerpo doctrinal en que se apoya en la actualidad la tecnología de simulación avanzada. En cuanto a la segunda, fueron sus

investigaciones las que condujeron al desarrollo de los programas comerciales de animación que hoy en día se utilizan, profusamente, para la creación de anuncios televisivos o secuencias de películas como Terminator II o Parque Jurásico.

El segundo de los campos de investigación, el de los periféricos sofisticados para conducción de aeronaves, fue evolucionando de manera paralela al de generación de gráficos 3-D. Numerosos centros civiles y militares como el MIT, la Universidad de Utah, la Universidad de Carolina del Norte y el Armstrong Aerospace Medical Research Laboratory Center sentaron las bases sobre las que muy pronto se construiría la industria de los periféricos de realidad virtual.

El primer prototipo de visiocasco fue construido en 1968 en Harvard por Ivan Sutherland (fundador más tarde de la compañía Evans & Sutherland) y David Cohen. Su nombre era la "Espada de Damocles". Ivan Sutherland había desarrollado como tesis doctoral en el MIT, a primeros de los sesenta, una serie de programas 2D interactivos utilizando el primer ordenador transistorizado, el TX-2. David Cohen, por su parte, había desarrollado el primer simulador de vuelo 3D, que utilizaba gráficos vectoriales.

La Espada de Damocles, desarrollada según una idea que el propio Sutherland había propuesto en su tesis doctoral tres años atrás, consistía en dos diminutos tubos de rayos catódicos (de media pulgada de diámetro) mediante los cuales podían contemplarse imágenes gráficas sobreimpresionadas en la escena real, gracias a un sistema de espejos. El dispositivo estaba suspendido del techo mediante un brazo mecánico (de ahí el nombre), lo que permitía conocer la posición y orientación de la cabeza del usuario. El primer prototipo era monoscópico, pero luego se añadió la capacidad de estereoscopía. Con aquel prototipo, el usuario podía ver la estructura de un objeto flotando en mitad de la habitación, pudiendo contemplar las distintas caras a medida que se desplazaba por la misma.

Con los criterios de hoy en día, la Espada de Damocles no era muy avanzada desde el punto de vista gráfico, pero para aquélla época se trataba de una auténtica revolución. De hecho, con aquel prototipo acababan de nacer los sistemas de realidad virtual. El sistema incluía ya prácticamente todos los conceptos en los que la tecnología de realidad virtual iba a basarse: gráficos tridimensionales, estereoscopía, localización de la posición del usuario, posibilidad de navegar alrededor de un objeto y contemplarlo desde todas las posiciones, técnicas de inmersión, etc, ...

Sutherland se trasladó en 1970 a la Universidad de Utah, donde continuaría su trabajo en la Espada de Damocles. Junto con David Evans, jefe del Departamento de Informática de la Universidad de Utah, Sutherland fundaría la empresa Evans&Sutherland, una de las líderes mundiales en estaciones de simulación avanzada. Por aquella universidad pasarían, en aquellos años, varios de los personajes que más adelante sentarían las bases de las industrias de animación, estaciones gráficas y realidad virtual. Así, el ya mencionado Ed Catmull, fundador de la empresa Pixar, o Jim Clark, quien fundó la empresa Silicon Graphics.

Algunos años más tarde, en 1977, se inició en el Armstrong Aerospace Medical Research Laboratory Center un proyecto militar, dirigido por Tom Furness, cuyo objetivo era estudiar la aplicación de los entornos virtuales a las cabinas de avión. Este proyecto daría como

resultado, cuatro años más tarde, la creación de un prototipo de visiocasco, que llevaba por nombre "Simulador de sistemas aéreos acoplado visualmente" (VCASS, Visually-Coupled Airborne Systems Simulator). Este casco, que realizaba también una superposición de gráficos sobre imágenes reales, era enorme y engorroso, pero fue el primero en incorporar tubos de rayos catódicos de alta resolución, así como un sistema de localización con seis grados de libertad para medir la posición y la orientación de la cabeza del usuario.

Pero los trabajos de investigación sobre periféricos avanzados, durante toda aquella década de los setenta, no se limitaron a los dispositivos de visualización. En 1970 se fundaba la compañía Polhemus, pionera en el campo de los dispositivos de localización electromagnética. En 1972, Frederick Brooks, de la Universidad de Carolina del Norte, desarrolla el primer prototipo de sistema con realimentación cinestésica. En 1975, Milton Krueger diseña un sistema donde se utilizaba el reconocimiento de imágenes para la detección de la posición del usuario. En 1977, Sandin y Sayre inventan en Chicago los primeros guantes utilizables como periféricos de entrada.

Al iniciarse la década de los ochenta, los avances en tecnología informática y microelectrónica dieron un nuevo impulso a la investigación sobre periféricos de realidad virtual y generación de gráficos 3-D, al abaratar las herramientas de proceso, al permitir la realización de herramientas software más complejas y al reducir los costes de implementación de circuitos integrados especializados en el tratamiento de imágenes. Lo que hasta ese momento había permanecido sumido en los canales de la investigación académica y militar, comenzó a despertar el interés de la industria.

Es en un laboratorio industrial donde vamos a encontrarnos con otro personaje de relevancia, Scott Fisher, quien está considerado (junto con Ivan Sutherland y Tom Furness) como uno de los "padres fundadores" de la realidad virtual. Fisher provenía del MIT, donde había estado comandando un grupo que trabajaba en el tema de pantallas estereoscópicas. En el año 1982 Scott Fisher comienza a trabajar para Atari.

En aquella época, el laboratorio de Atari, dirigido por Alan Kay, era un auténtico campo de experimentación de futuras tecnologías. El problema es que, por lo general, si la investigación no se dedica a sacar productos a plazo no muy lejano, resulta ruinosa. Al final, el laboratorio tuvo que cerrar, pero fue un vivero en el que se formaron algunos de los que, más tarde, jugarían un papel destacado en el desarrollo de la realidad virtual. Allí, por ejemplo, Fisher coincidió con Eric Gullichsen, quien después de una temporada en Autodesk fundaría Sense8, compañía líder en el campo de las herramientas de desarrollo de aplicaciones de RV. También estaba allí William Bricken, quien también trabajó para Autodesk antes de marchar al Human Interface Technology Laboratory (HITL) de la Universidad de Washington.

Scott Fisher, después de su paso por el laboratorio de Atari, se incorporaría al NASA AMES Center en 1986 para trabajar en el proyecto VIEW. Por aquella época, el NASA Ames Center, en California, estaba dirigido por David Nigel. Allí, Michael McGreevy y Jim Humphries habían construido en 1985 un visiocasco estereoscópico de bajo coste, denominado VIVED (Virtual Visual Environment Display). El VIVED fue desarrollado para la preparación de misiones espaciales a partir de elementos disponibles comercialmente,

integrando dos pantallas LCD y un sensor de posición Polhemus. Los LCDs eran de una televisión de bolsillo, sobre los que pusieron unas lentes LEEP de gran angular. El proyecto VIVED evolucionaría más tarde para convertirse en el Virtual Interface Environment Workstation (VIEW), proyecto al que se unirían en 1986 Scott Fisher y W. Robinett. Fue en el marco de este proyecto donde muchos de los conceptos que forman parte de los actuales paradigmas de control de aplicaciones inmersivas (escalado del mundo virtual, vuelo virtual, manipulación de objetos, etc...) fueron desarrollados.

Por todas partes empiezan a surgir equipos de desarrollo trabajando en la tecnología de la RV y, lo que es más importante, se empiezan a ver los primeros resultados comerciales. En 1980, la compañía StereoGraphics inventa las gafas de visión estéreo. Dos años después, en 1982, Thomas Zimmerman patenta el Dataglove, un electroguante que inventó mientras investigaba sobre cómo controlar con la mano un instrumento musical virtual. El electroguante empezaría a ser comercializado cuatro años más tarde por la empresa californiana VPL Research. En 1987, la compañía inglesa Dimension International (ahora denominada Superscape) desarrolla un software de construcción de mundos tridimensionales sobre PC, denominado Freescape. En 1988, Scott Foster, de Crystal River Engineering, inventa el Convolvotron, dispositivo para la generación de sonido tridimensional. Ese mismo año J. Waldern crea en Inglaterra W Industries, que en 1993 pasaría a ser el Virtuality Group, y Mark Bolas funda Fake Space Labs, empresa que fabrica sistemas binoculares para RV. También en 1988 AutoDesk funda su laboratorio de investigación sobre realidad virtual, llamado Autodesk Cyberspace Project, más tarde Cyberia.

Al año siguiente, 1989, Atari saca al mercado la primera máquina de galería de videojuegos con tecnología 3D. Ese mismo año, LEEP Systems y VPL comienzan a fabricar los primeros visiocascos comerciales; Lusted y Knapp fundan BioControl Systems, dedicada al desarrollo de dispositivos de control bioeléctrico; Tom Furness deja el ejército para fundar el Human Interface Technology Laboratory (HITL) en la Universidad de Washington, al que se incorporaría William Bricken, procedente de Autodesk. También en 1989, Eric Gullichsen funda Sense8, y Autodesk presenta su primer sistema de realidad virtual para PC.

A partir de aquí, entramos de lleno en la carrera comercial. Los sistemas de RV comienzan a popularizarse, y una miríada de productos empiezan a invadir el mercado.

En paralelo, alrededor de la realidad virtual comienza a configurarse un cierto movimiento cultural (o contracultural, si se prefiere) conocido como cyberpunk y surgido alrededor de la literatura del mismo nombre.

El nombre de cyberpunk está sacado del título de un cuento corto de Bruce Berthke, y fue Gardner Dozois, director de la Isaac Asimov's Science Fiction Magazine, el primero en utilizarlo para referirse a un tipo particular de literatura de ciencia ficción.

El argumento de las primeras novelas cyberpunk giraba alrededor del tradicional héroe marginal que se rebela contra el sistema social, pero con dos variantes tecnológicas: el Sistema es una especie de dictadura tecnológica, que cuenta con avanzados medios técnicos para oprimir a los ciudadanos (es decir, es un Tecnosistema), y el héroe suele también ayudarse en su lucha de sofisticadas herramientas, sean éstas consolas de ordenador ultramodernas, dispositivos de realidad virtual o prótesis cibernéticas.

Figura 8.1. Fotograma de la película Blade Runner.
(Fotografía de The Blade Runner Partnership)

Alrededor de este punto de partida se desarrolló toda una mitología en la que la marginalidad, las escenas urbanas, los ambientes sombríos, las sociedades hipertecnificadas, la realidad virtual, los robots y los ordenadores se funden en una subcultura (dicho sin ánimo peyorativo) en la que los avances técnicos son a la vez elementos de opresión y herramientas de liberación, y en la que no necesariamente el héroe es el bueno.

El elemento de referencia es, sin lugar a dudas, la novela Neuromante, de William Gibson, publicada en 1985 y que ganó los premios Hugo y Nebula. Neuromante es el primer libro de una trilogía que completan Cuenta Cero (1986) y Mona Lisa Overdrive (1988). William Gibson sería, precisamente, el primero en utilizar la palabra ciberespacio, en su cuento Burning Chrome, donde se hace extensivo uso de los conceptos de realidad virtual.

A partir de ahí, el movimiento cyberpunk cobró cuerpo y fama, hasta que la propia palabra ha llegado a ser de uso común. Publicaciones como Mondo 2000 y Wired han contribuido a la difusión de la cultura asociada con el cyberpunk, y a su alrededor se ha ido fraguando buena parte del movimiento.

En la actualidad, la palabra cyberpunk se usa de modo casi común para designar a cualquier grupo de marginales de la informática. Katie Hafner y John Markoff publicaron, en 1991, un libro con el título de Cyberpunk, en el que hacen una recopilación de los más famosos casos de piratería informática de los últimos años. En otro orden de cosas, la música

tecno y la música industrial suelen considerarse como música cyberpunk. Billy Idol ha sacado recientemente al mercado un disco con ese título precisamente: Cyberpunk.

La estética y temática cyberpunk, y sus derivados, han impregnado también la TV y el cine. Al lado de ejemplos de calidad más bien discutible, como Max Headroom o El cortador de césped, muchas buenas películas guardan una relación más o menos cercana con este movimiento, como Alien, Terminator, Desafío Total o una de las mejores películas de todos los tiempos, y la que tal vez mejor resume la estética del cyberpunk: Blade Runner, de Ridley Scott.

8.2. LA REALIDAD VIRTUAL HOY

8.2.1. Panorama del sector

El mercado de la realidad virtual está evolucionando rápidamente, desde sus orígenes militares y académicos, hacia el mundo de los sistemas comerciales. Esta evolución se ha plasmado en la aparición de multitud de productos comerciales de RV, que han permitido que las aplicaciones de bajo coste de RV sobre PC sean ya un hecho.

Hay numerosas compañías que ofrecen los productos necesarios para integrar un sistema de realidad virtual completo (periféricos de RV, tarjetas gráficas, sistemas de desarrollo para aplicaciones de RV, ...) a un coste muchísimo menor que el que era habitual hace tres años. Y los costes continúan descendiendo, a medida que la tecnología continúa madurando y aparecen compañías de nueva creación compitiendo por un hueco en el mercado. Gracias a ello, el mercado de RV está empezando a expandirse de manera acelerada, estando ya prácticamente en condiciones de llegar al mercado de consumo.

En el sector de los periféricos, son las compañías americanas las que predominan de manera indiscutible: Polhemus y Ascension Technologies en lo referente a dispositivos de localización electromagnéticos; Logitech en el campo de los ratones 3D; VPL y Virtual Research en el de los visiocascos; Fake Space Labs en el de los sistemas binoculares; StereoGraphics en el de las gafas estereoscópicas; Crystal River Engineering en el campo de los dispositivos de sonido 3D; VPL, Exos y Virtual Technologies en el campo de los electroguantes; BioControl Systems en el de los sistemas de control bioeléctrico, ...

En el sector de las herramientas de desarrollo de aplicaciones de RV, dos compañías británicas, Superscape y Division, compiten con dos americanas, Sense8 y VREAM, por el control del mercado, con IBM intentando activamente penetrar en el sector.

En el de las plataformas, podemos encontrar cuatro tipos diferentes:

- Equipos profesionales de simulación, donde destacan compañías como la americana Evans & Sutherland o la británica Division.
- Estaciones gráficas de propósito general, principalmente empleadas en la industria publicitaria, pero que también juegan un papel preponderante en la construcción de simuladores y sistemas de RV, donde la compañía americana Silicon Graphics es el líder indiscutido del mercado.

- Equipos de RV sobre PC que utilizan aceleradores gráficos para proporcionar la necesaria velocidad de síntesis de las escenas, donde destacan la compañía alemana Spea y la británica Division, aunque las compañías americanas Matrox y Kubota comienzan a hacer su aparición.
- Sistemas propietarios para aplicaciones de ocio, donde destaca la empresa británica Virtuality.

Como principales foros sobre RV podemos destacar la conferencia SIGGRAPH, que se celebra anualmente en EEUU, y la revista CyberEdge Journal, que mensualmente publica un repaso de los últimos avances y acontecimientos en el mundo de la RV.

En cuanto a las aplicaciones, el sector de los juegos continúa demostrando que es, con mucha diferencia, el mayor de los que componen el mercado potencial. Sólo como ejemplo, el juego Sonic II de Sega recaudó 500 millones de dólares en 1993 en los Estados Unidos, 200 millones de dólares más que la renombrada película Parque Jurásico. La tecnología de tratamiento de gráficos 3D encontrará, previsiblemente, su mayor mercado en el campo de los videojuegos de galería y las consolas de juegos de uso doméstico.

La evolución previsible apunta a:

- Un cercano boom de la RV sobre PC, que ha motivado que empresas como Kubota abandonen los desarrollos sobre estaciones Alpha para concentrarse en el mercado del PC, y que Silicon Graphics lance su iniciativa Open GL, para tratar de controlar el desarrollo de nuevas herramientas software y aceleradores gráficos.
- Una convergencia rápida con el mercado de los videojuegos. Las compañías japonesas, que carecían de la tecnología de generación de gráficos 3D, se han apresurado a establecer alianzas con empresas activas en dicho campo (Namco con Evans & Sutherland, Nintendo con Silicon Graphics y Sega con Division). De hecho, los fabricantes de sistemas de generación de gráficos se pueden convertir en proveedores de la tecnología, y las compañías japonesas tradicionales adoptar el papel de meros diseñadores de juegos.
- Una explosión de los sistemas de RV en red, tanto sobre área local, como sobre cable, como sobre línea telefónica. Sega, ATT, Microsoft y una miríada de actores comienzan a explorar las posibilidades que abre la oferta de servicios de valor añadido sobre línea telefónica con soporte gráfico y soporte multimedia.

8.2.2. La RV en España y en Europa

Como nota sorprendente, el campo de la RV no es uno en los que las compañías europeas estén más rezagadas. Aunque la mayoría de las compañías que operan en el sector son americanas, Europa cuenta con varios actores de primera fila: las británicas Superscape y Division en el campo de los sistemas de desarrollo de aplicaciones de RV, la alemana Spea en el de los aceleradores gráficos para PC, y la británica Virtuality, líder absoluto en el mercado de sistemas inmersivos para ocio. Las tres primeras compañías operan, además, precisamente en el segmento de mercado que más crecimiento va a experimentar en los próximos años, que es el de la RV sobre PC.

Además de estas compañías, muchas otras empresas y organismos de investigación europeos operan en el sector, permitiendo albergar ciertas esperanzas en cuanto a la evolución futura del mercado de la RV en Europa. Así, la empresa francesa Caption, del grupo Telmat, que comercializa las estaciones gráficas de gama alta TWS y circuitos integrados especializados para el tratamiento de gráficos; las también francesas Simulis y Sextant Avionique en el sector de los visiocascos; la noruega Stereovisjon en el campo de los sistemas autoestereoscópicos; la francesa Virtools en el de los joysticks 3D; la Escuela Superior Santa Ana de Pisa y el Trinity College de Dublín en el de los electroguantes; el CIRAD francés en el terreno del software para la creación de modelos tridimensionales de plantas, ...

En España, no existen muchas empresas que trabajen en el campo de la RV, pero a pesar de todo sí existen algunos ejemplos. Entre ellos podemos citar:

- La empresa APD, que lidera un consorcio de empresas europeas, en el marco del programa ESPRIT, para el desarrollo de sistemas para el mercado de los juegos 3D y de realidad virtual. En dicho proyecto, que recibe el nombre de MOONLIGHT, participan, además de APD, el Trinity College de Dublín, el Norwegian Computing Center, las empresas francesas Caption y Opsis, la empresa belga Deltatec, y la alemana Nova Apparate. En el proyecto MOONLIGHT se están desarrollando tanto subsistemas para el tratamiento de gráficos 3D (sobre PC y otras arquitecturas), como software de desarrollo de aplicaciones de RV en red.
- Construcciones Aeronáuticas (CASA), que participa en un proyecto de la ESA (European Space Agency) donde se está desarrollando un visiocasco de alta resolución para entrenamiento de astronautas.
- CESELSA, compañía que cuenta con una amplia experiencia en el desarrollo de simuladores de vuelo.
- RTZ Technologies, que además de ostentar la representación de Superscape en España, desarrolla aplicaciones de RV.
- El LISITT (Laboratorio Integrado de Sistemas Inteligentes y Tecnologías de la Información en Tráfico) de Valencia, donde a mediados de 1992 se creó un grupo de investigación y desarrollo de aplicaciones de informática gráfica y simulación en tiempo real, y en especial de entornos de realidad virtual. Dicho grupo es, actualmente, uno de los que más activamente trabajan dentro de este campo. A finales de 1992 presentó, en la feria VIDEGRAF-MERCAFILM de Valencia, un entorno interactivo en tiempo real representando el interior y contenidos de la futura Ciudad de las Ciencias de Valencia. Ha desarrollado asimismo un simulador de conducción (SIRCA), aprovechando la experiencia existente en el laboratorio sobre ingeniería de tráfico.
- Realidad Virtual S.L., empresa pionera dentro del campo de la realidad virtual en España, que ha contribuido grandemente a popularizar este tipo de tecnología en nuestro país gracias a la organización de exposiciones interactivas por toda la geografía nacional, por las que han pasado ya más de 140.000 personas. Esta empresa organiza asimismo cursos y conferencias sobre realidad virtual, realiza labores de consultoría dentro de este campo y ha desarrollado también diversas aplicaciones dentro del campo de la publicidad, el ocio y la industria.

8.3. LÍNEAS PRINCIPALES DE INVESTIGACIÓN

En los próximos tres años, como ya hemos mencionado, el mercado de la RV sufrirá una rápida evolución hacia el mercado de consumo, debido al abaratamiento de los componentes de los sistemas de RV.

Pero las líneas de investigación no se centran exclusivamente en el tema de reducción de los costes. El mercado de la RV se encuentra actualmente en una situación similar a la que el mercado de los ordenadores experimentó en los últimos años de la década de los setenta: los costes comenzaban a hacer posible la popularización de la informática, pero aún hubo de transcurrir algún tiempo hasta que se llegó a definir el producto que habría de hacer explotar el mercado. No era sólo cuestión de precio, sino de definir las características adecuadas de interfaz, de prestaciones, de aplicaciones, ...

En estos momentos, nadie es capaz todavía de predecir el aspecto final que el mercado de la RV va a tener. Esta incertidumbre viene provocada por dos factores.

El primero es la tecnología: ¿qué tecnología va a estar disponible y cuándo? No se trata sólo de averiguar si una determinada solución va a estar disponible en una fecha concreta, sino también de saber de cuánto tiempo se dispone a partir de ese momento, antes de que dicha tecnología se vuelva obsoleta. Desde el punto de vista tecnológico, dos son los campos que están ejerciendo una mayor presión sobre el mercado: la tecnología de gráficos 3-D y los avances software en orientación a objetos y arquitecturas cliente-servidor.

El segundo (y principal) factor de incertidumbre es el usuario: ¿qué tipo de aplicaciones, productos y servicios aceptará (y demandará) el usuario? A lo largo de los últimos años, son muchos los ejemplos que muestran que el usuario medio no se deja impresionar por las meras consideraciones tecnológicas. La expectación despertada por una tecnología no se traduce necesariamente en volumen de facturación. A menos que los productos satisfagan una cierta necesidad del usuario, la tecnología es completamente inútil, por muy avanzada que sea.

En este apartado vamos a repasar brevemente las principales líneas en que se centra la investigación actual sobre sistemas de RV, señalando las perspectivas existentes, y los principales problemas planteados.

8.3.1. Periféricos

8.3.1.1. Visiocascos

En el campo de los dispositivos de visualización, los esfuerzos se centran en dos líneas diferentes. Por un lado, en la obtención de pantallas LCD con mayor resolución y menor coste. Las soluciones actuales, si bien perfectamente aptas para la experimentación, no proporcionan aún la calidad suficiente como para poder utilizar texto con un mínimo de resolución, lo cual limita grandemente el campo de aplicación.

Por otro lado, los sistemas inmersivos, como ya se comentó en el capítulo 3, presentan entre sus desventajas el hecho de que el usuario está imposibilitado para usar instrumentos complejos, dado que no puede verlos. Además, el hecho de emplear una interfaz en tres

dimensiones vuelve obsoletas las metáforas bidimensionales utilizadas en los equipos informáticos actuales (menús, ventanas). En consecuencia, existe un considerable interés en la investigación sobre nuevos paradigmas de control y mecanismos de interfaz para sistemas inmersivos, sin que por el momento parezca existir ninguna línea clara de avance.

Con respecto al tema de la resolución, el Departamento de Defensa americano está desarrollando una serie de proyectos que permitirán, en 3 años, disponer de pantallas LCD color para visiocasco de alta resolución. En Europa también se está desarrollando tecnología para la fabricación de visiocascos de alta resolución, dentro del proyecto patrocinado por la ESA (European Space Agency) anteriormente mencionado, en el que participa la empresa española Construcciones Aeronáuticas.

En paralelo, se está procediendo también con la investigación sobre nuevas tecnologías de visualización, alternativas a los LCDs. Entre las iniciativas en este sentido, cabe señalar el Visual Retinal Display que está siendo desarrollado por la Universidad de Washington. Con el Visual Retinal Display, las imágenes se reproducen directamente en la retina del usuario mediante un rayo láser, lo que permite alcanzar resoluciones de hasta 4000x4000 pixeles. El sistema está todavía en fase experimental; los principales problemas son, en este momento, el alto volumen del dispositivo y el hecho de que no se dispone de láseres azules, lo cual elimina uno de los tres canales de color.

8.3.1.2. Dispositivos de localización

En el terreno de los dispositivos de localización, la investigación sigue también dos líneas diferentes. Por un lado, la investigación continúa para el desarrollo de dispositivos electromagnéticos con mayor rango de acción, menor error y menor latencia, dado que los actuales son bastante deficientes en estos aspectos.

Por otro lado, diversas empresas y equipos de investigación están explorando la posibilidad de emplear tecnologías alternativas, como el reconocimiento de imágenes, los dispositivos inerciales o los compases electrónicos.

8.3.1.3. Electroguantes

Los desarrollos en el campo de los electroguantes se centran en los aspectos prácticos de disminución de costes y mejora de la fiabilidad, para lo cual se están buscando alternativas a las tecnologías existentes.

Pero es el campo del software el que más interés despierta en estos momentos, en relación a este tipo de dispositivos de control. Todavía no se han hallado paradigmas de control adecuados que permitan hacer un uso extensivo de los electroguantes en aplicaciones de RV. La principal línea de investigación apunta a los lenguajes gestuales (el reconocimiento automático de gestos efectuados con las manos); en este campo, hay que destacar las investigaciones llevadas a cabo por el Trinity College de Dublín en el proyecto GLAD-IN-ART, dentro del marco del programa europeo ESPRIT. Queda todavía trabajo por hacer,

sin embargo, hasta lograr que el reconocimiento de gestos tenga el mismo grado de fiabilidad que tiene hoy en día el reconocimiento de voz.

8.3.1.4. Otros dispositivos

Por supuesto, cabe esperar que durante los próximos años aparezcan nuevos dispositivos de control tridimensional (ratones, joysticks, controladores de fuerza/giro) de bajo coste, así como que se termine de popularizar el uso de sistemas de reconocimiento de voz, especialmente en aplicaciones de tele-robótica.

También cabe esperar avances considerables en el campo de los sistemas de control bioeléctricos. Por un lado, se está investigando en el refinamiento de los dispositivos de detección del electro-óculo-grama, de forma que pueda detectarse no sólo la dirección de visión, sino también la profundidad a la que el usuario dirige su mirada. De esa manera, los sistemas de RV serían capaces de lidiar con el fenómeno de la convergencia y la acomodación, lo que permitiría una visión estereoscópica más realista y evitaría los problemas de carácter visual que los dispositivos actuales presentan.

Por otro lado, distintas empresas y organismos están investigando activamente sobre el tema de identificación de ondas cerebrales, aunque es imposible predecir cuánto tiempo pasará antes de que seamos capaces de controlar un sistema de RV mediante comandos subvocalizados.

8.3.2. Dispositivos gráficos

El desarrollo referente a dispositivos gráficos se centra actualmente en las siguientes direcciones:

- Desarrollo de nuevos circuitos integrados especializados en el tratamiento de gráficos 3D. Esto incluye tanto nuevos circuitos de procesamiento ráster, más rápidos, como dispositivos capaces de realizar parte de los cálculos correspondientes a la etapa de procesamiento geométrico. Este último aspecto tropieza con el inconveniente de que, mientras que la fase de rasterización es más o menos común a todas las aplicaciones de tratamiento de gráficos 3D, la fase de procesamiento geométrico es altamente dependiente de la aplicación. Como consecuencia, al fijar en hardware parte de este proceso, se limita el campo de aplicación de los correspondientes circuitos integrados.
- Desarrollo de nuevas tarjetas aceleradoras para el entorno PC y nuevos módulos gráficos para las estaciones de gama alta y el mercado de videojuegos. El desarrollo de este tipo de tarjetas y módulos depende en primera instancia de la disponibilidad de los nuevos circuitos integrados a los que hemos hecho referencia, así como de los avances en cuanto a módulos de memoria de alta velocidad de acceso. Actualmente, uno de los cuellos de botella en los módulos de aceleración gráfica radica en la velocidad de acceso a la memoria de video y a la memoria de z-buffering. Durante los próximos años cabe esperar una disminución de los costes de las memorias de video y las memorias estáticas de alta velocidad, lo que permitirá incrementar la velocidad de proceso en este tipo de tarjetas.

No se prevén avances espectaculares en cuanto a los algoritmos empleados en la síntesis de imágenes, los cuales son relativamente maduros. Sin embargo, existe una incertidumbre en cuanto al tipo de interfaz que acabará imponiéndose para los aceleradores gráficos, especialmente en el mercado del PC. El problema radica actualmente en saber si se acabará generalizando una interfaz de bajo nivel tipo Open GL o si, por el contrario, terminarán adoptándose distintas interfaces de alto nivel, cada una de ellas enfocada a un segmento particular del mercado.

8.3.3. Paradigmas de control

Posiblemente sea en el campo del software donde quepa esperar los avances más espectaculares, porque es en el terreno de los sistemas operativos, herramientas de desarrollo y lenguajes donde la RV actual se encuentra más necesitada.

Son varios los frentes abiertos a este respecto. En primer lugar se halla el tema de los paradigmas de control, a que ya antes hemos hecho referencia. Las aplicaciones 3D son esencialmente diferentes de las aplicaciones tradicionales, donde la pantalla actúa casi como una mera hoja de papel electrónica. El carácter tridimensional de las aplicaciones abre toda una serie de nuevos interrogantes en cuanto a cómo navegar en un entorno virtual, cómo manipular objetos, cómo organizar la información, cómo representar gráficamente los conceptos abstractos, ...

Muchas preguntas permanecen aún sin respuesta acerca de sobre qué bases construir un mundo virtual: ¿por qué los usuarios muestran mayor tendencia a "perderse" en un mundo virtual que en el mundo real?, ¿qué leyes físicas y lógicas del mundo real es necesario respetar, y cuáles pueden ser alteradas sin que la efectividad del operador se resienta? La respuesta a estas preguntas, de carácter más psicológico que tecnológico, es necesaria si se quiere llegar a definir el producto capaz de abrir a la RV las puertas del mercado de gran consumo.

8.3.4. Comportamiento e interacción

El segundo de los frentes se refiere al problema del comportamiento e interacción entre objetos dentro de un mundo virtual, problema al que ya hemos hecho referencia en un capítulo anterior. Los entornos de ejecución actuales están bien adaptados a la simulación de "larga distancia", donde los objetos que interactúan en un mundo virtual se encuentran muy separados unos de otros (como sucede en un simulador de vuelo o de tanques, por ejemplo). Pero la mayoría de las aplicaciones profesionales y de uso doméstico requieren una simulación de "corta distancia", donde exista un mayor grado de interacción entre los objetos, al estilo de lo que sucede en los videojuegos. En el campo del diseño arquitectónico, por ejemplo, la "larga distancia" inherente a los sistemas de RV hoy en día disponibles se traduce en que el usuario sólo puede interactuar con el entorno de una forma bastante limitada.

Para que la realidad virtual se convierta en un mercado significativo, es necesario resolver el problema de la interacción. Esto significa proporcionar medios para dotar de comportamientos complejos a los objetos que forman el mundo virtual y, como consecuencia,

asociar a los objetos complejas secuencias de animación que se correspondan con el comportamiento descrito.

Pero esto es, precisamente, lo que hoy en día puede encontrarse en el sector de los juegos: mundos virtuales en los que el usuario interactúa con objetos cuyo comportamiento puede ser extremadamente complejo, y en los que las secuencias gráficas se emplean para mostrar al usuario los efectos de dicho comportamiento. La única diferencia entre el sector de la realidad virtual y el de los juegos es de carácter tecnológico: mientras que la realidad virtual trata de abrirse camino desde el mundo de los gráficos 3D hacia el mundo de la interactividad, el sector de los juegos se desplaza en la dirección contraria, tratando de incorporar la tecnología de tratamiento de gráficos 3D a sus productos interactivos.

Dada la rápida convergencia que se está produciendo entre el mercado de la RV y el de los videojuegos, cabe esperar que la asimilación tecnológica que está teniendo lugar fructifique en los próximos años, y aparezcan nuevos entornos de ejecución, nuevas herramientas de desarrollo de aplicaciones y nuevos paradigmas de definición de mundos virtuales.

Es bastante posible que los requerimientos impuestos por las aplicaciones de RV obliguen a nuevos desarrollos en el campo de los lenguajes de programación. Dada la complejidad que este tipo de aplicaciones requiere, a la hora de definir e implementar el comportamiento de los objetos que forman parte de una simulación, cada vez se está extendiendo más el empleo de lenguajes orientados a objeto. Los lenguajes orientados a objeto, como Smalltalk, Eiffel o C++ ofrecen facilidades para la programación de sistemas complejos que no existen en otro tipo de lenguajes.

Sin embargo, ni siquiera esas facilidades son suficientes. El diseño de aplicaciones de simulación a gran escala plantea problemas aún no resueltos por los lenguajes orientados a objeto actuales, relativos a la herencia múltiple de comportamientos, la utilización de objetos activos, la exposición de interfaces en tiempo de ejecución (que atenta contra el propio concepto de encapsulación en que los lenguajes orientados a objetos actuales se basan) y la interacción funcional entre objetos. Para muchas de estas cuestiones, como la de la herencia múltiple de comportamientos en objetos activos, ni siquiera existen respuestas a nivel teórico. Por eso comienza a abrirse paso el concepto de la programación orientada a actores, que trataría de superar los inconvenientes que la programación orientada a objetos presenta a la hora de desarrollar aplicaciones de simulación a gran escala.

Entre las iniciativas dirigidas a resolver las cuestiones derivadas de este tipo de problemas podemos citar el desarrollo del sistema operativo VEOS (Virtual Environment Operating System), por parte del Human Interface Technology Laboratory de la Universidad de Washington, y los trabajos realizados dentro del proyecto europeo MOONLIGHT, liderado por la empresa española APD.

Con respecto al tema de las herramientas de desarrollo, los problemas actuales de carácter teórico se centran en encontrar paradigmas para la definición del comportamiento de los objetos que permitan la reutilización de código y la composición del comportamiento. En términos simples, podríamos enunciar el problema de la forma siguiente: si queremos modelar el comportamiento de un objeto complejo, la única manera de hacerlo es descom-

poner dicho comportamiento en varios sub-comportamientos simples, que luego puedan componerse de forma sencilla. Sin embargo, este problema de enunciado tan simple carece por el momento de soluciones prácticas.

8.3.5. Sistemas de RV en red

El tercero de los frentes es el de los sistemas de RV en red (ciberespacios). Las previsiones de mercado indican que, sea cual sea el aspecto final del "producto estrella" que pueda abrir el mercado de consumo para los sistemas de RV, se tratará de un producto dotado de posibilidades de conexión en red.

Pero aplicar mecanismos de distribución en los sistemas de RV plantea problemas de carácter técnico nada desdeñables. En el último apartado de este capítulo expondremos más en detalle el estado actual de la cuestión. Baste decir que, en la actualidad, no es que no existan productos comerciales capaces de responder a las expectativas que el mercado de ciberespacios abre: es que ni siquiera existe cuerpo de doctrina teórico para muchos de los interrogantes planteados.

8.4. APLICACIONES DE MAÑANA

La aceleración a que el mercado de la RV se ha visto sometido en el último año está teniendo su reflejo en la multitud de nuevas aplicaciones que cada día surgen. Aunque el campo de aplicación de la RV está aún limitado por las carencias de los actuales sistemas operativos, a que antes hemos hecho referencia, en los próximos dos años podemos esperar que la tendencia se acelere aún más.

No es mi intención, con este apartado, realizar un análisis exhaustivo de los campos de aplicación posibles, lo cual sería bastante prolijo, pero sí al menos reseñar algunos de ellos, donde se prevé un fuerte crecimiento a corto plazo.

8.4.1. Aplicaciones médicas

El campo de la medicina es, tal vez, uno de los que más prometedores parecen de cara al futuro de la realidad virtual, aunque todavía quedan algunos años antes de poder aplicar la RV en el uso clínico. La RV encuentra múltiples áreas de aplicación en el campo de la medicina, entre las cuales podemos destacar la enseñanza y entrenamiento quirúrgico y la visualización médica.

Con respecto a la enseñanza y el entrenamiento quirúrgico, una vez que se resuelvan satisfactoriamente los problemas asociados a la simulación de resistencias, fricciones, etc... los sistemas de RV con realimentación cinestésica permitirían efectuar operaciones virtuales sobre pacientes virtuales, con un alto grado de realismo. Esto serviría tanto para entrenar a nuevos cirujanos, como para enseñar nuevas técnicas a los cirujanos en ejercicio. Varias compañías están, en estos momentos, desarrollando diversos sistemas de entrenamiento quirúrgico. Techplanations, por ejemplo, está desarrollando un simulador quirúrgico para laparoscopias. Georgia Tech, por su parte, está desarrollando un simulador con realimenta-

ción cinestésica para cirugía ocular. Medical Media Systems, en fin, está desarrollando un sistema virtual para entrenamiento en cirugía endoscópica.

En el campo de la visualización médica, las imágenes obtenidas por ultrasonidos, resonancia magnética o tomografía computerizada pueden emplearse en aplicaciones de RV que pongan herramientas de visualización más completas a disposición de los médicos. Dicha "reconstrucción virtual" del interior de pacientes reales puede emplearse no sólo a efectos diagnósticos, sino también, por ejemplo, para la planificación de intervenciones quirúrgicas.

8.4.2. Sistemas de ocio

El más grande crecimiento a corto plazo se encuentra en el campo de los sistemas de ocio. El número de compañías trabajando en el desarrollo de nuevos juegos inmersivos y de plataforma se multiplica por momentos, y los simuladores de gama alta para el mercado doméstico ya comienzan a hacer su aparición.

Las compañías de videojuegos continuarán, por su parte, su evolución hacia los sistemas con gráficos 3D, y es de esperar una rápida popularización de videojuegos de galería con dispositivos autoestereoscópicos.

En el mercado doméstico, la tendencia iniciada hace dos años hacia el desarrollo de videoconsolas con tecnología 3D continuará, aunque no está muy claro cuáles serán las características finales del producto que se acabe imponiendo. La primera compañía en sacar al mercado una videoconsola dotada de capacidad de generación de gráficos 3D fue la americana 3DO, que contaba con el apoyo de grandes compañías de dentro y fuera del sector de los juegos, como Electronic Arts, Matshushita, Time Warner, Panasonic y ATT. La experiencia de 3DO no ha sido, sin embargo, excesivamente satisfactoria hasta el momento.

Atari sacó al mercado a fines de 1993 su consola Jaguar, diseñada por Atari y fabricada por IBM, que aunque emplea un procesador de 16 bits, dispone de dos chips especializados de tratamiento gráfico (llamados Tom y Jerry) para la generación de gráficos 3-D, con sombreado de Gouraud, z-buffering, texturado y 24-bits por color.

Sega está desarrollando una nueva consola de 64 bits, cuyo nombre clave es Saturn, que utilizará un procesador RISC de Hitachi actualmente en desarrollo. Los planes iniciales eran sacar la consola al mercado a finales de 1994, aunque finalmente parece que la consola saldrá al mercado a lo largo de 1995. Al parecer, el sistema operativo está siendo diseñado por Microsoft.

Nintendo, por su parte, ha firmado un acuerdo con Silicon Graphics para el desarrollo de una consola de 64 bits con posibilidad de generación de gráficos 3-D. La consola utilizará un procesador RISC MIPS. El proyecto tiene el nombre de Project Reality. Los planes iniciales eran sacar las primeras máquinas de galería basadas en esta tecnología en 1994, y sacar las primeras consolas a finales de 1995, con el nombre de Ultra 64.

La cuestión de si estos sistemas evolucionarán hacia auténticas plataformas de RV está aún por decidir. Los primeros intentos de emplear tecnología de periféricos sofisticada, como la comercialización por parte de Mattel de su electroguante de bajo coste PowerGlove, no

han tenido el éxito esperado, y nadie sabe la respuesta que el mercado de consumo tendrá ante los nuevos visiocascos de bajo coste que comienzan a aparecer.

En cualquier caso, la popularización de las consolas con tecnología 3D contribuirá a la aparición de nuevos chips de bajo coste para el tratamiento de este tipo de gráficos, lo que indirectamente incidirá en el crecimiento del mercado de la RV sobre PC.

8.4.3. Sistemas de entrenamiento y simulación

Dentro del campo de los equipos de simulación, los sistemas de carácter profesional (simuladores de vuelo, sistemas de entrenamiento para policía y bomberos, simuladores de carácter militar) continuarán representando la parte más importante del mercado durante algún tiempo, pero ya comienzan a hacer su aparición los primeros sistemas orientados a mercados más populares.

Por un lado, ya han comenzado a salir al mercado los primeros simuladores de carácter deportivo, para ciclismo, golf, esquí e incluso béisbol, que incorporan parte de las tecnologías empleadas en los actuales sistemas de RV.

Por otro lado, el abaratamiento de costes hace prever la pronta generalización de simuladores de conducción para autoescuelas, lo que puede provocar bastantes cambios en el sector.

8.5. RV EN RED Y CIBERESPACIOS

8.5.1. Sistemas de RV en red

Una de las principales tendencias que se apunta en el horizonte de la RV es la de la evolución hacia sistemas de RV distribuida y las aplicaciones de ciberespacio.

En los capítulos anteriores hemos centrado nuestra exposición en los sistemas monousuario, donde se utiliza un único dispositivo de presentación. En este apartado, vamos a hacer un repaso del estado actual de los sistemas de RV en red, así como la previsible evolución futura.

Antes de comenzar, conviene clarificar algunos de los conceptos:

8.5.1.1. Ciberespacios

Llamamos **sistema de RV multiusuario o ciberespacio** a aquél donde varios usuarios, utilizando cada uno su propio dispositivo de presentación comparten un mismo mundo virtual, en el cual puede ocupar cada uno una posición distinta. En un ciberespacio, por tanto, cada uno de los usuarios tiene una visión del mundo virtual diferente (debido a su distinta posición), y el sistema debe ser capaz de presentar a cada uno la correspondiente imagen de la escena. A todos los efectos, un ciberespacio es un conjunto de sistemas de RV

diferentes, cada uno con sus propias entradas y salidas, pero que comparten la base de datos de la escena y forman parte de la misma simulación.

8.5.1.2. Sistemas multiparticipante

Llamamos **sistema de RV multiparticipante** a aquél sistema donde varios usuarios comparten un mismo mundo virtual, pero utilizando *el mismo dispositivo de presentación*. Es el caso, por ejemplo, de los sistemas proyectivos donde varios usuarios se introducen en una cápsula de simulación. En realidad, el sistema desconoce la existencia de esos múltiples usuarios. Para él, existe una única "ventana" al mundo virtual, que corresponde a la cápsula como un todo, independientemente de cuántos usuarios la ocupen. A todos los efectos es como si, en un sistema monousuario de sobremesa, varios usuarios se pusieran al mismo tiempo a contemplar la pantalla: no por eso dejará de ser monousuario.

8.5.1.3. Sistemas distribuidos

Llamamos **sistema de RV distribuido** a aquél donde se emplean varias estaciones de proceso diferentes. Hay que aclarar que ni todo ciberespacio tiene por qué ser distribuido, ni todo sistema distribuido es necesariamente un ciberespacio. Como ya hemos comentado en el capítulo 6, al hablar del tema de la estación de proceso, las arquitecturas distribuidas también juegan un importante papel a la hora de configurar un sistema de RV monousuario. Utilizar arquitecturas pipeline o arquitecturas paralelas permite aumentar la capacidad total de proceso del sistema, distribuyendo las tareas entre distintos ordenadores, por lo que se las utiliza ampliamente para configurar sistemas donde no hay más que un usuario simultáneo.

De la misma manera, nada nos impide construir un sistema de RV donde no haya más que una estación de proceso, pero con diferentes dispositivos de presentación y mandos de control, de manera que cada usuario pueda interactuar con el sistema por separado. En este caso tendríamos un ciberespacio centralizado; este tipo de sistema no es habitual, pero sería perfectamente factible desde el punto de vista técnico: la estación de proceso tendría simplemente que atender a tantos conjuntos de dispositivos de entrada, y generar tantas imágenes de la escena, como usuarios haya.

8.5.1.4. Sistemas en red

Llamamos **sistema de RV en red** a todo sistema multiusuario donde cada usuario disponga de su propia estación de proceso, es decir, un ciberespacio distribuido.

En lo sucesivo, vamos a usar indistintamente los términos "ciberespacio" y "sistema de RV en red".

8.5.2. Implementación de los sistemas de RV en red

La implementación de un ciberespacio es simple desde el punto de vista conceptual. Básicamente, lo que hacemos es multiplicar el número de "ventanas" al mundo virtual y el número de dispositivos de control empleados.

Si antes el sistema (considerado en su conjunto) tenía que calcular la imagen que del mundo virtual tenía el usuario, ahora hay que hacer lo mismo, pero repitiendo el proceso para cada uno de los múltiples usuarios. Si antes había que tener en cuenta, para realizar la simulación, los datos recibidos de una serie de dispositivos de control, ahora habrá también que hacer lo mismo, con la única diferencia de que el número de dispositivos de control distintos se habrá multiplicado por el número de usuarios.

Existen dos casos bien diferenciados de ciberespacio, según el tipo de red de comunicaciones que se utilice para conectar a las distintas estaciones de proceso: sistemas de baja latencia y sistemas de alta latencia.

8.5.2.1. Arquitecturas de baja latencia

Cuando conectamos las distintas estaciones de proceso mediante una red de área local, la comunicación entre las mismas es relativamente rápida: el tiempo que transcurre desde

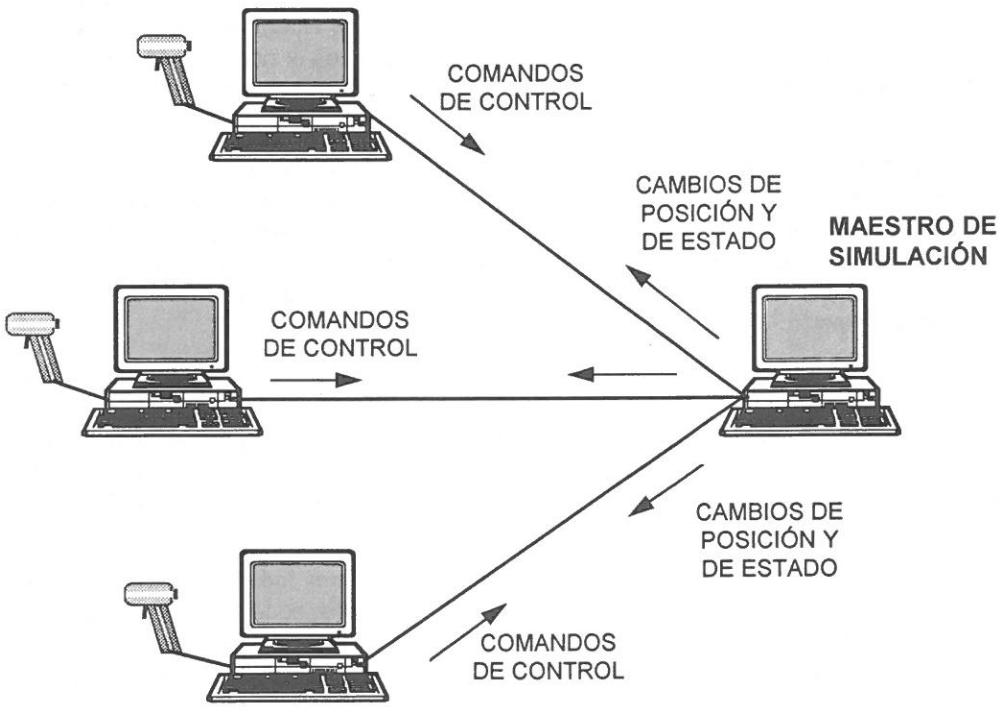

Figura 8.2. Sistema de baja latencia con simulación centralizada

que una de las estaciones envía un conjunto de datos a otra, hasta que ésta lo recibe, es relativamente corto (del orden de los milisegundos o, como mucho, decenas de milisegundos). Por eso este tipo de sistema se conoce con el nombre de sistema de baja latencia.

En los sistemas de baja latencia, la solución más simple consiste en realizar una simulación centralizada, utilizando una de las estaciones de proceso como "maestro de simulación", y distribuir todas las tareas de entrada y salida. La figura 8.2 muestra un ejemplo de este tipo de sistema.

En este tipo de sistema, cada una de las estaciones se encarga de recibir e interpretar las órdenes que su correspondiente usuario emite a través de los dispositivos de control. Dichas órdenes son transmitidas por las estaciones a través de la red hacia el maestro de simulación, en forma de comandos de alto nivel.

La estación maestra realiza la simulación del mundo virtual, a partir de los comandos de control que le llegan de las estaciones esclavas y, como resultado, actualiza el estado y las posiciones de los distintos objetos dentro del mundo virtual. Estos cambios de posición y de estado son transmitidos entonces a las estaciones esclavas, las cuales se encargan de sintetizar la escena *en modo local*, de acuerdo con la posición de su correspondiente usuario.

Como se verá, las tareas de localización, síntesis de la escena y el resto de las interacciones con el usuario se llevan a cabo sin que la estación maestra tenga que intervenir. Si un usuario mueve su ratón 3D por la escena, tan sólo se entera del hecho su estación esclava correspondiente. Sólo cuando el usuario seleccione con el ratón un cierto objeto, la estación esclava emitirá un mensaje de alto nivel, informando a la maestra de que su usuario ha efectuado una selección.

Todas las estaciones comparten una misma base de datos de la escena, a la cual acceden para poder sintetizar la imagen y efectuar el resto de las operaciones de salida hacia el usuario. De esa base de datos puede haber una única copia (situada en la estación maestra) o varias copias idénticas (una en la maestra y otra en cada esclava), pero la única que puede efectuar cambios en el contenido de la base de datos es la estación maestra, como resultado de cada ciclo de simulación.

Para aquellos que estén familiarizados con X-Windows y los entornos Unix, la situación es parecida a cuando tenemos una aplicación X (un X-Client) corriendo en una máquina y comunicándose con varios terminales X (varios X-Server) distintos. La aplicación se encarga de implementar toda la funcionalidad, y los terminales llevan a cabo todas las tareas de entrada y salida, y la comunicación entre una y otros se realiza mediante un protocolo de alto nivel (el X-Protocol).

Obsérvese que, con una arquitectura de este estilo, hemos independizado la tarea de simulación de los detalles concretos de la interfaz. Si queremos, podemos hacer coexistir en un mismo sistema estaciones esclavas de distinto tipo: con visiocasco, de sobremesa, con o sin sonido 3D, etc... Dependiendo de los dispositivos periféricos que tenga asociados, cada estación esclava llevará a cabo unas tareas u otras, sin que ello afecte al maestro de simulación.

Hay que aclarar que la topología de red en estrella que aparece en la figura es sólo ilustrativa, para señalar que la comunicación tiene lugar entre el maestro y cada una de las estaciones esclavas, pero cualquier otra topología de red es igualmente válida. También hay que señalar que la estación maestra no tiene por qué ser diferente en su constitución de las esclavas: podría ser perfectamente cualquiera de las estaciones esclavas la que actúe como maestra en un sistema dado; lo único que hay que garantizar es que la estación que actúe como maestra disponga de la capacidad de proceso suficiente para efectuar, además de su propia entrada y salida, la simulación de toda la red.

En este tipo de arquitecturas puede perfectamente avanzarse un paso más en la distribución, y repartir entre todas las estaciones las tareas de simulación, de forma que cada una se encargue de efectuar la simulación de un subconjunto de los objetos.

Esta solución es, sin embargo, más complicada que la de simulación centralizada, ya que surgen dos problemas diferentes:

- El primero es el de las **interacciones entre objetos**. Si dos objetos cuya simulación se lleva a cabo en un mismo nodo interactúan, no hay problema alguno; el problema surge cuando se produce una interacción (por ejemplo una colisión) entre objetos controlados por nodos diferentes. En este caso, alguien tiene que ser el responsable final de decidir qué ha ocurrido con la interacción.

  Como ejemplo, si una bala está siendo controlada por un nodo, y su objetivo por otro, alguien tiene que tomar la decisión de si la bala alcanzó el objetivo o no. ¿Quién es el que decide si se produjo la colisión: la estación que controla la bala o la que está simulando el comportamiento del objetivo? Una solución frecuentemente empleada consiste en mantener centralizadas aquellas partes del mecanismo de simulación que tienen que ver con las interacciones, por ejemplo definiendo un Gestor de Colisiones único para toda la red.

- El segundo problema es el de la **sincronización en las actualizaciones de la base de datos**. Cuando existe un único proceso dedicado a la simulación, realizar las actualizaciones del estado del sistema es sencillo, pero cuando hay varios procesos de simulación ejecutándose en paralelo, cada uno de ellos puede modificar el estado del sistema en cualquier momento. En consecuencia, se hace necesario implementar mecanismos que garanticen que esas actualizaciones se realizan de manera consistente (por ejemplo utilizando transacciones).

  Sin esos mecanismos, varios procesos podrían intentar actualizar simultáneamente un mismo conjunto de datos, produciéndose resultados impredecibles. El problema se complica aún más por el hecho de que, al estar la simulación distribuida, la base de datos representativa del estado del sistema suele también ser distribuida.

La comunicación entre nodos en este tipo de escenario se realiza mediante un protocolo de alto nivel, con el que los nodos se intercambian la información referente a los cambios de posición y de estado de los objetos que cada uno controla (ver figura 8.3).

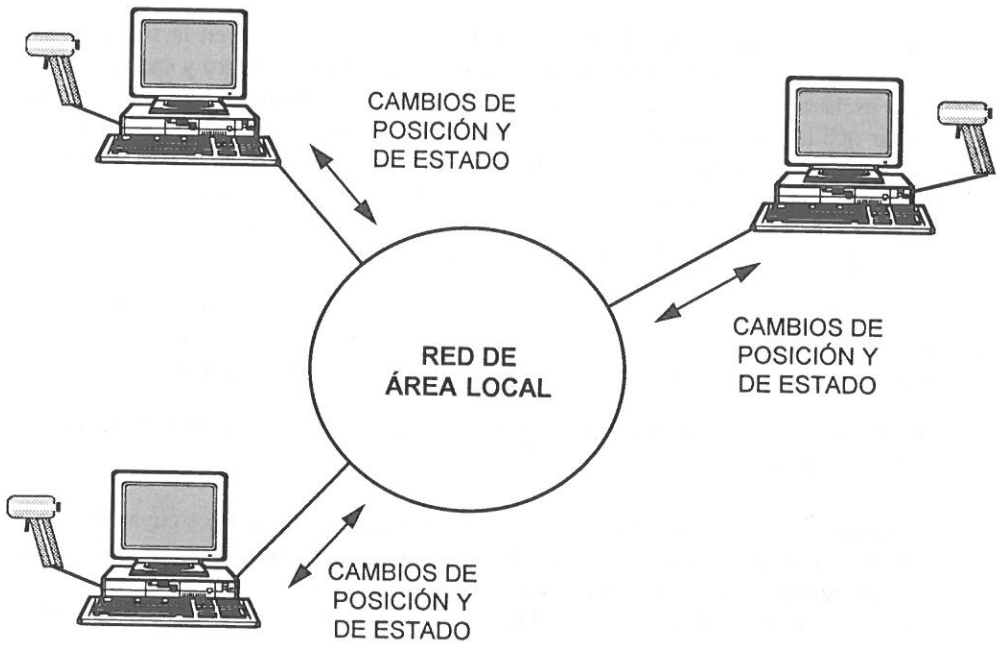

Figura 8.3. Sistema de baja latencia con simulación distribuida

8.5.2.2. Arquitecturas de alta latencia

Como hemos podido ver, cuando la red de comunicaciones introduce una latencia pequeña, se puede recurrir a un mecanismo de simulación centralizada, con lo que la implementación de un ciberespacio no presenta excesivos desafíos técnicos. Sin embargo, la cosa se complica cuando la red posee una latencia alta, es decir, cuando los mensajes intercambiados entre unas estaciones de proceso y otras tardan en llegar un tiempo no despreciable. Esto es lo que sucede, por ejemplo, en sistemas montados sobre redes de área extendida o red telefónica de conmutación.

En este caso, el mantener un maestro central de simulación hace que los tiempos de respuesta a las acciones del usuario sean muy elevados, lo cual precluye cualquier interfaz de carácter inmersivo. A pesar de ello, la simulación centralizada sigue teniendo la gran ventaja de la simplicidad, por lo que ésta es la solución adoptada en sistemas como Habitat, del que más abajo hablaremos.

La solución opuesta, la de emplear mecanismos de simulación distribuida, también se ve afectada por los retardos de la red. En el caso de redes de baja latencia, cuando un objeto cambia de posición, el nodo correspondiente informa al resto de los nodos, y éstos reciben la información con un retardo muy pequeño, por lo cual la posición se actualiza de forma

prácticamente simultánea en toda la red. En el caso de una figura humana desplazándose, por ejemplo, el nodo responsable de controlarla manda un mensaje por cada uno de los pequeños incrementos de posición que el desplazamiento implica.

Pero cuando los retardos de comunicación son grandes, el mensaje tarda en llegar al resto de los nodos y, además, esos retardos no son constantes, sino que pueden variar de uno a otro mensaje debido, por ejemplo, a colisiones en la red. En consecuencia, en el caso de la figura humana desplazándose por la pantalla, los nodos no pueden quedarse esperando a que les llegue cada una de las actualizaciones de posición, porque eso daría lugar a movimientos "espasmódicos" de los objetos y, además, lo que el usuario vería estaría retardado con respecto a la posición "real" del objeto dentro del mundo virtual.

En consecuencia, en los sistemas de este estilo (como el NPSNET que más abajo describimos), se recurre a métodos de estimación (métodos de dead-reckoning), mediante los cuales cada nodo "estima" en todo momento la posición de los objetos que no controla, a partir de los últimos datos conocidos de posición y velocidad de dicho objeto. Periódicamente, cada nodo envía información al resto de los nodos sobre la posición de los objetos que controla, para que las estimaciones no se desvíen demasiado de la posición real.

Obsérvense las consecuencias de esto: dado un cierto objeto, la información acerca de la posición que dicho objeto ocupa dentro del mundo virtual es diferente en cada nodo de la red. Esto trae consigo algunas limitaciones, como a continuación se detalla.

8.5.2.3. Interacción próxima y lejana

Los sistemas de alta latencia presentan un inconveniente adicional, que limita su campo de aplicación. En cualquier sistema de simulación, sistema de RV o videojuego, existen interacciones entre los objetos que podríamos calificar de interacciones "próximas", en el sentido de que los objetos que interactúan están cerca unos de otros.

Por ejemplo, cuando un usuario "agarra" un objeto con su mano virtual, el objeto y la mano están adyacentes dentro del mundo virtual. Una pequeña diferencia de posición (por ejemplo si la mano está algo más alejada del objeto) puede hacer que el usuario no sea capaz de agarrar el objeto, al considerar el proceso responsable de la simulación que no está lo suficientemente cerca.

Este tipo de sistema, donde las pequeñas diferencias de posición pueden tener un efecto significativo en la simulación, recibe el nombre de **sistemas de interacción próxima**.

En las arquitecturas en red de alta latencia con simulación distribuida se pueden producir diferencias significativas en la información de posición de un objeto, entre unos y otros nodos de la red. Lo cual implica que no es posible implementar sistemas de interacción próxima mediante una arquitectura de este estilo. Un ejemplo nos ayudará a ver por qué:

- Supongamos que hay un objeto desplazándose por el mundo virtual, controlado por un cierto nodo A, y que un usuario, en otro nodo B, decide en un momento determinado colocar su mano (virtual) delante del objeto, para detenerlo. En el nodo A, la información de posición del objeto es la real, pero la de la mano es estimada. En el

nodo B sucede exactamente al contrario. En consecuencia, puede haber pequeñas discrepancias entre la información de posición de ambos nodos y es posible que el nodo B "estime" que el objeto ha chocado con la mano (debiendo por tanto detenerse), mientras que el nodo A "estima" que la mano no logró detener al objeto.

Por consiguiente, si queremos evitar este tipo de inconsistencias, la única manera de implementar un sistema de interacción próxima es recurriendo a simular de manera centralizada al menos todos los aspectos que involucren interacción entre objetos situados en nodos distintos.

En el caso de los **sistemas de interacción lejana**, esto es, sistemas donde los errores de posición no tienen una influencia tan grande sobre la simulación, puede perfectamente emplearse tanto una simulación centralizada como distribuida.

Este es el caso, por ejemplo, de los simuladores distribuidos de batallas terrestres empleados por el ejército americano (redes SIMNET y NPSNET, que más abajo se describen). En este tipo de simuladores, las distancias entre objetos (por ejemplo entre dos tanques) son altas, por lo que no tiene ninguna importancia el que existan pequeñas diferencias en la información de posición entre un nodo y otro.

Existe un estándar específico del IEEE, el Std 1278, que trata de los protocolos empleados para las aplicaciones de simulación interactiva distribuida de carácter militar sobre red de alta latencia. En el estándar se describen los mecanismos de comunicación y estimación de posiciones utilizados en este tipo de sistema de interacción lejana.

8.5.3. Algunos ejemplos de ciberespacios actuales

8.5.3.1. SIMNET Y NPSNET

El Pentágono americano, preocupado por los problemas de las maniobras militares a gran escala que se hacían en Europa (alto coste monetario, problemas políticos, etc...) encargó a la Universidad de Florida Central el desarrollo de un simulador distribuido para 200 tanques, la red SIMNET. Cada nodo era un simulador de un tanque M-1, con sus controles reales, capaz de llevar a una tripulación de cuatro personas, y un total de 8 pantallas situadas en las ventanillas del tanque. El sistema era 2-D, con una resolución muy pobre (320x138), por lo que su catalogación como sistema de RV era bastante discutible, pero el sistema resultaba tremendamente efectivo como método de entrenamiento, como la Guerra del Golfo vendría a corroborar.

La generación siguiente de simuladores militares distribuidos, como la red NPSNET, ya puede realmente considerarse un sistema de RV en toda regla.

NPSNET ha sido desarrollado en la Escuela Naval de Posgraduados del ejército americano, y permite efectuar simulaciones de combate terrestre y aéreo entre múltiples usuarios. Utiliza estaciones Silicon Graphics, y admite varios tipos de interfaz, por lo que en una misma red pueden coexistir estaciones con distintos tipos de dispositivo de visuali-

zación: estaciones con visiocasco, estaciones con sistema binocular, sistemas de sobremesa con gafas estereoscópicas, o sistemas monoscópicos convencionales.

El sistema está basado en el estándar IEEE 1278. Cada nodo de la red controla una serie de objetos, para los cuales realiza la simulación correspondiente. Cada nodo va estimando asimismo, de modo constante, la posición de los objetos controlados por los otros nodos. Estas estimaciones emplean los algoritmos de dead-reckoning establecidos en el estándar citado, y que a continuación se describen.

Imaginemos, por ejemplo, que el nodo A está realizando la simulación de movimiento de un tanque T_A, y el nodo B de otro tanque T_B. El nodo A mantiene un algoritmo de simulación completo, con el que calcula los cambios de posición del tanque T_A, y otro modelo simplificado, con el que estima las posiciones del tanque T_B a partir de los últimos datos conocidos de posición y velocidad del mismo. Cada vez que el tanque T_B cambia de dirección o de velocidad, el nodo B se lo comunica al A, para que éste ajuste sus estimaciones.

Para evitar que las estimaciones se alejen demasiado de la posición real de los objetos, el nodo B también tiene una copia del algoritmo simplificado que A usa para estimar la posición del tanque T_B. A la hora de actualizar la posición de su tanque, el nodo B utiliza el algoritmo completo, pero también aplica el algoritmo simplificado para averiguar cuál es la posición que A va a estimar. Si la estimación se aleja demasiado de la posición real, B manda un mensaje al nodo A comunicándole la posición del tanque T_B, para que el nodo A ajuste sus estimaciones.

Independientemente de lo anterior, todos los nodos mandan una información completa de posición de sus objetos cada pocos segundos, con el fin de que todo el mundo ajuste sus estimaciones sin esperar a que las discrepancias crezcan demasiado.

El software de NPSNET ha sido realizado en C y C++, usando como API el Performer de Silicon Graphics, y como base de datos Multigen. La comunicación entre nodos se realiza sobre UDP, utilizando los protocolos del estándar IEEE 1278.

8.5.3.2. Habitat

Habitat, diseñado para LucasFilm por Chip Morningstar y Randall Farmer, fue uno de los primeros intentos (y desde luego el de más éxito) de crear un entorno gráfico comercial multiusuario a gran escala.

Habitat fue diseñado como una ciudad virtual en la que los usuarios estaban representados por una figura animada en dos dimensiones, que recibía el nombre de "avatar". Los avatares podían caminar por la ciudad, trasladándose de una escena a otra, coger determinados objetos, realizar transacciones con el dinero virtual que circulaba en la ciudad, participar en diversos juegos e interactuar con los avatares correspondientes a los otros usuarios.

La ciudad constaba de más de 20.000 escenas diferentes, dispuestas según una rejilla cuadrangular. Para pasar de una escena a otra, el usuario no tenía más que desplazar su avatar hasta el límite de la pantalla.

Para mover las figuras, el usuario empleaba un joystick, y para hablar con otros usuarios el mensaje se escribía en el teclado y aparecía encima de la cabeza de la figura en forma de "bocadillo" (como en las viñetas de los tebeos).

Habitat era un servicio fundamentalmente conversacional, dado que el principal atractivo radicaba en la interacción con otros usuarios. La metáfora de dibujos animados utilizada para la interfaz gráfica dotaba de poderoso atractivo al servicio, pero la esencia de éste era fundamentalmente la de un servicio de mensajería. No se proporcionaban servicios de carácter transaccional (comercial) dentro de la ciudad virtual, y los juegos disponibles eran bastante limitados (debido, básicamente, a las limitaciones técnicas del sistema).

Los usuarios accedían al sistema desde ordenadores Commodore 64, en los que se ejecutaba un programa que era el responsable de generar las imágenes de la ciudad, de acuerdo con la información de estado que el centro servidor enviaba a los terminales. Por otro lado, dicho programa se encargaba de traducir las acciones del usuario (movimientos del joystick, mensajes introducidos mediante el teclado) a comandos de alto nivel que se enviaban al centro servidor.

El centro servidor era el encargado de realizar la simulación, es decir, de mantener la base de datos descriptiva de la ciudad virtual, y actualizarla de acuerdo con las acciones que los usuarios realizaban (se usaba, por tanto, simulación centralizada). Cada acción de un usuario (por ejemplo un movimiento) era transmitida a aquéllos otros que, por estar situados en la misma escena, podían verse afectados por la misma. De esa manera se aseguraba la consistencia de la visión que cada usuario tenía de la ciudad virtual.

Los terminales se comunicaban con el centro servidor a 300 bps. La red sobre la que funcionaba el sistema era QuantumLink, un servicio on-line comercial sobre red de conmutación de paquetes. El retardo que la red imponía a cada paquete transmitido desde el centro servidor al terminal, y viceversa, oscilaba entre los 100 mseg y los 5 segundos.

Esto da una idea, primero, de las limitaciones que el sistema tenía. De hecho, el gran retardo existente en las comunicaciones limitaba las posibilidades de creación de juegos de "acción". Pero, por otro lado, deja ver cómo la creación de un ciberespacio no requiere tecnología sofisticada.

Después de una fase experimental, el sistema ha pasado a ser comercial, con el nombre de Club Caribe. En la actualidad, después de cinco años de funcionamiento, cuenta con más de 15.000 abonados. Una versión japonesa, diseñada por Fujitsu, lleva tres años operando sobre la red NIFtyServe. Presenta una funcionalidad igual a la de su antecesor americano, aunque la interfaz gráfica y de sonido es bastante mejor, al utilizar como terminales los ordenadores Fujitsu FM Towns, de más altas prestaciones.

La principal lección de un sistema como Habitat es que lo importante, en un ciberespacio, son las capacidades de interacción que se ponen a disposición de los usuarios: interacción tanto con el sistema como con otros usuarios. Lo que a éstos les importa es qué pueden hacer dentro del ciberespacio, es decir, qué servicios (conversacionales, transaccionales, lúdicos) el ciberespacio pone a su disposición. Las limitaciones técnicas, como el retardo de respuesta, tienen (hasta cierto punto) una importancia menor.

TÍTULOS PUBLICADOS EN LA COLECCIÓN 99%

AUTOCAD

AUTOSKETCH

CLIPPER

COMUNICACIONES. Interfaces. Modems. Redes. Normas y Protocolos

dBASE III PLUS

dBASE IV

DISCO DURO

DISPLAYWRITE 4

ENTORNO PROGRAMADOR DEL OPEN ACCESS II + OPEN ACCESS II PLUS

EXCEL 4 PARA WINDOWS

EXCEL 5

FICHEROS GRÁFICOS. Interconexión entre aplicaciones

FRAMEWORK

HARVARD GRAPHICS PARA WINDOWS

HYPERCARD E HYPERTALK. Hipertexto para Macintosh

IBM-PC COMPATIBLES

IMPRESORAS MATRICIALES

INTERNET

KNOSYS

LOTUS FREELANCE PLUS

LOTUS 1-2-3

LOTUS 1-2-3. Versión 3

LOTUS 1-2-3. PARA WINDOWS

MACROS EN LOTUS 1-2-3

MATHEMATICA

MICROSOFT ACCESS

MODELADO SÓLIDO CON AUTOCAD 12

MODEMS. Todo sobre telecomunicaciones

MULTIMATE 4

NOVEDADES WORDPERFECT 6

OPEN ACCESS II

OPEN ACCESS III y IV

OrCAD SDT/III

ORCAD PCB II

ORCAD VST. Verificación y simulación de circuitos digitales por ordenador

PAGE MAKER PARA WINDOWS

PC-Tools de luxe. Versión 6

POSTCRIPT

PROGRAMACIÓN ORIENTADA AL OBJETO EN CLIPPER 5.01

PROGRAMACIÓN Y ANÁLISIS ESTADÍSTICOS BÁSICOS CON SPSS/PC(+)

PSPICE. Simulación y Análisis de Circuitos

QUATTRO PRO 5

QUATTRO PRO PARA WINDOWS

SERIE ASSISTANT DE IBM

SQL

STORYBOARD PLUS

SUPERCARD. Más allá del Hypercard

SYMPHONY

TÉCNICAS AVANZADAS CON LOTUS 1-2-3

3D STUDIO. Animación y proyectos de Diseño

3D STUDIO. Diseño de formas planas 2 de shaper

TRES DIMENSIONES CON AUTOCAD 12

UTILIDADES NORTON

VENTURA

WINDOWS NT 3.1

WINDOWS 3.1

WINDOWS 3.1 para trabajos en grupo

WORD 5. Tratamiento de textos

WORDPERFECT 5.1

WORDPERFECT 5.2 PARA WINDOWS

WORDPERFECT PRESENTATIONS

WORDPERFECT 6

WORDPERFECT 6 PARA DOS

WORDPERFECT 6 PARA WINDOWS

WORDSTAR

WORDSTAR 6

WORKS. Versión 2 en castellano